Computing Principal Components Dynamically

Darko Dimitrov [1], Mathias Holst [2], Christian Knauer [1], Klaus Kriegel [1]

[1] Institut für Informatik, Freie Universität Berlin,
Takustraße 9, D–14195 Berlin, Germany
{darko,knauer,kriegel}@inf.fu-berlin.de
[2] Institute of Computer Science, Universität Rostock,
Albert Einstein Str. 21, D-18059 Rostock, Germany
mholst@informatik.uni-rostock.de

Abstract

In this paper we present closed-form solutions for efficiently updating the principal components of a set of n points, when m points are added or deleted from the point set. For both operations performed on a discrete point set in $\mathbb{R}^d$, we can compute the new principal components in $O(m)$ time for fixed d. This is a significant improvement over the commonly used approach of recomputing the principal components from scratch, which takes $O(n+m)$ time. An important application of the above result is the dynamical computation of bounding boxes based on principal component analysis. PCA bounding boxes are very often used in many fields, among others in computer graphics for collision detection and fast rendering. We have implemented and evaluated few algorithms for computing dynamically PCA bounding boxes in $\mathbb{R}^3$. In addition, we present closed-form solutions for computing dynamically principal components of continuous point sets in $\mathbb{R}^2$ and $\mathbb{R}^3$. In both cases, discrete and continuous, to compute the new principal components, no additional data structures or storage are needed.

1 Introduction

Principal component analysis (PCA) [15] is probably the oldest and best known of the techniques of multivariate analysis. The central idea and motivation of PCA is to reduce the dimensionality of a point set by identifying *the most significant directions (principal components)*. Let $P = \{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_n\}$ be a set of vectors (points) in $\mathbb{R}^d$, and $\vec{\mu} = (\mu_1, \mu_2, \ldots, \mu_d) \in \mathbb{R}^d$ be the center of gravity of P. For $1 \leq k \leq d$, we use $p_{i,k}$ to denote the k-th coordinate of the vector p_i. Given two vectors $\vec{u}$ and $\vec{v}$, we use $\langle \vec{u}, \vec{v} \rangle$ to denote their inner product. For any unit vector $\vec{v} \in \mathbb{R}^d$, the *variance of P in direction $\vec{v}$* is

$$\mathrm{var}(P, \vec{v}) = \frac{1}{n} \sum_{i=1}^{n} \langle p_i - \vec{\mu}\,,\, \vec{v} \rangle^2. \tag{1}$$

The most significant direction corresponds to the unit vector $\vec{v}_1$ such that $\mathrm{var}(P, \vec{v}_1)$ is maximum. In general, after identifying the j most significant directions $\vec{v}_1, \dots, \vec{v}_j$, the $(j+1)$-th most significant direction corresponds to the unit vector $\vec{v}_{j+1}$ such that $\mathrm{var}(P, \vec{v}_{j+1})$ is maximum among all unit vectors perpendicular to $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_j$.

It can be verified that for any unit vector $\vec{v} \in \mathbb{R}^d$,

$$\mathrm{var}(P, \vec{v}) = \langle \Sigma\, \vec{v}, \vec{v} \rangle, \tag{2}$$

where Σ is the *covariance matrix* of P. Σ is a symmetric $d \times d$ matrix where the (i,j)-th component, $\sigma_{ij}, 1 \le i, j \le d$, is defined as

$$\sigma_{ij} = \frac{1}{n} \sum_{k=1}^{n} (p_{i,k} - \mu_i)(p_{j,k} - \mu_j). \tag{3}$$

The procedure of finding the most significant directions, in the sense mentioned above, can be formulated as an eigenvalue problem. If $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_d$ are the eigenvalues of Σ, then the unit eigenvector $\vec{v}_j$ for λ_j is the j-th most significant direction. All λ_js are non-negative and $\lambda_j = \mathrm{var}(X, \vec{v}_j)$. Since the matrix Σ is symmetric positive definite, its eigenvectors are orthogonal.

Computation of the eigenvalues, when d is not very large, can be done in $O(d^3)$ time, for example with the *Jacobi* or the *QR method* [18]. For very large d, the problem of computing eigenvalues is non-trivial. In practice, the above mentioned methods for computing eigenvalues converge rapidly. In theory, it is unclear how to bound the running time combinatorially and how to compute the eigenvalues in decreasing order. In [10] a modification of the *Power method* [17] is presented, which can give a guaranteed approximation of the eigenvalues with high probability.

Examples of many applications of PCA include data compression, exploratory data analysis, visualization, image processing, pattern and image recognition, time series prediction, detecting perfect and reflective symmetry, and dimension detection. The thorough overview over PCA's applications can be found for example in the textbooks [13] and [15]. Most of the applications of PCA are non-geometric in their nature. However, there are also few purely geometric applications that are quite spread in computer graphics. Example are the estimation of the undirected normals of the point sets or computing PCA bounding boxes (bounding boxes determined by the principal components of the point set).

Dynamic versions of the above applications, i.e., when the point set (population) changes, are of big importance and interest. In this paper we present closed-form solutions for efficiently updating the principal components of a dynamic point set. We also consider the computation of the dynamic PCA bounding boxes - a very important application in many fields including computer graphics, where the PCA boxes are used to maintain hierarchical data structures for fast rendering of a scene or for collision detection.

Based on the theoretical results in this paper, we have implemented several algorithms for computing PCA bounding boxes dynamically.

The organization and the main results of the paper are as follows: In Section 2 we present closed-form solutions for efficiently updating the principal components of a set of

n points, when m points are added or deleted from the point set. For both operations performed on a discrete point set in $\mathbb{R}^d$, we can compute the new principal components in $O(m)$ time for fixed d. This is a significant improvement over the commonly used approach of recomputing the principal components from scratch, which takes $O(n+m)$ time. In Section 3 we consider solutions for the static and dynamic versions of the bounding box problem. In Section 4 we present and verify the correctness of the theoretical results presented in the Chapter 2. We have implemented several dynamic PCA bounding box algorithms and evaluated their performances. Conclusion and open problems are presented in Section 5. In the appendix we consider the computation of the principal components of a dynamic continuous point set. We give closed form-solutions when the point set is a convex polytope or a boundary of a convex polytope in $\mathbb{R}^2$ or $\mathbb{R}^3$. When the point set is a boundary of a convex polytope, we can update the new principal components in $O(k)$ time, for both deletion and addition, under the assumption that we know the k facets in which the polytope changes. Under the same assumption, when the point set is a convex polytope in $\mathbb{R}^2$ or $\mathbb{R}^3$, we can update the principal components in $O(k)$ time after adding points. But, to update the principal components after deleting points from a convex polytope in $\mathbb{R}^2$ or $\mathbb{R}^3$ we need $O(n)$ time. This is due to the fact that after a deletion the center of gravity of the old convex hull (polyhedron) could lie outside the new convex hull, and therefore, a retetrahedralization is needed (see Subsection 6.1.1 and Subsection 6.2.1 for details).

2 Updating the principal components efficiently - discrete case in $\mathbb{R}^d$

In this subsection, we consider the problem of updating the covariance matrix Σ of a discrete point set $P = \{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_n\}$ in $\mathbb{R}^d$, when m points are added or deleted from P. We give closed-form solutions for computing the components of the new covariance matrix Σ'. Those closed-form solutions are based on the already computed components of Σ. The main result of this section is given in the following theorem.

Theorem 1 *Let P be a set of n points in $\mathbb{R}^d$ with known covariance matrix Σ. Let P' be a point set in $\mathbb{R}^d$, obtained by adding or deleting m points from P. The principal components of P' can be computed in $O(m)$ time for fixed d.*

Proof.

Adding points

Let $P_m = \{\vec{p}_{n+1}, \vec{p}_{n+2}, \ldots, \vec{p}_{n+m}\}$ be a point set with center of gravity $\vec{\mu}^m = (\mu_1^m, \mu_2^m, \ldots, \mu_d^m)$. We add P_m to P obtaining new point set P'. The j-th component, μ'_j, $1 \leq j \leq d$, of the center of gravity $\vec{\mu}' = (\mu'_1, \mu'_2, \ldots, \mu'_d)$ of P' is

$$\begin{aligned}\mu'_j &= \tfrac{1}{n+m}\textstyle\sum_{k=1}^{n+m} p_{k,j} \\ &= \tfrac{1}{n+m}\left(\textstyle\sum_{k=1}^{n} p_{k,j} + \sum_{k=n+1}^{n+m} p_{k,j}\right) \\ &= \tfrac{n}{n+m}\mu_j + \tfrac{m}{n+m}\mu_j^m.\end{aligned}$$

The (i,j)-th component, σ'_{ij}, $1 \le i,j \le d$, of the covariance matrix Σ' of P' is

$$\begin{aligned}\sigma'_{ij} &= \tfrac{1}{n+m}\textstyle\sum_{k=1}^{n+m}(p_{k,i}-\mu'_i)(p_{k,j}-\mu'_j) \\ &= \tfrac{1}{n+m}\textstyle\sum_{k=1}^{n}(p_{k,i}-\mu'_i)(p_{k,j}-\mu'_j) + \tfrac{1}{n+m}\sum_{k=n+1}^{n+m}(p_{k,i}-\mu'_i)(p_{k,j}-\mu'_j).\end{aligned}$$

Let

$$\sigma'_{ij} = \sigma'_{ij,1} + \sigma'_{ij,2},$$

where,

$$\sigma'_{ij,1} = \frac{1}{n+m}\sum_{k=1}^{n}(p_{k,i}-\mu'_i)(p_{k,j}-\mu'_j), \tag{4}$$

and

$$\sigma'_{ij,2} = \frac{1}{n+m}\sum_{k=n+1}^{n+m}(p_{k,i}-\mu'_i)(p_{k,j}-\mu'_j). \tag{5}$$

Plugging-in the values of μ'_i and μ'_j in (4), we obtain:

$$\begin{aligned}\sigma'_{ij,1} &= \tfrac{1}{n+m}\textstyle\sum_{k=1}^{n}(p_{k,i}-\tfrac{n}{n+m}\mu_i-\tfrac{m}{n+m}\mu_i^m)(p_{k,j}-\tfrac{n}{n+m}\mu_j-\tfrac{m}{n+m}\mu_j^m) \\ &= \tfrac{1}{n+m}\textstyle\sum_{k=1}^{m}(p_{k,i}-\mu_i+\tfrac{m}{n+m}\mu_i-\tfrac{m}{n+m}\mu_i^m)(p_{k,j}-\mu_j+\tfrac{m}{n+m}\mu_j-\tfrac{m}{n+m}\mu_j^m) \\ &= \tfrac{1}{n+m}\textstyle\sum_{k=1}^{n}(p_{k,i}-\mu_i)(p_{k,j}-\mu_j) + \tfrac{1}{n+m}\sum_{k=1}^{n}(p_{k,i}-\mu_i)(\tfrac{m}{n+m}\mu_j-\tfrac{m}{n+m}\mu_j^m)+ \\ &\quad \tfrac{1}{n+m}\textstyle\sum_{k=1}^{n}(\tfrac{m}{n+m}\mu_i-\tfrac{m}{n+m}\mu_i^m)(p_{k,j}-\mu_j)+ \\ &\quad \tfrac{1}{n+m}\textstyle\sum_{k=1}^{n}(\tfrac{m}{n+m}\mu_i-\tfrac{m}{n+m}\mu_i^m)(\tfrac{m}{n+m}\mu_j-\tfrac{m}{n+m}\mu_j^m).\end{aligned}$$

Since $\sum_{k=1}^{n}(p_{k,i}-\mu_i)=0$, $1 \le i \le d$, we have

$$\sigma'_{ij,1} = \tfrac{n}{n+m}\sigma_{ij} + \tfrac{nm^2}{(n+m)^3}(\mu_i-\mu_i^m)(\mu_j-\mu_j^m). \tag{6}$$

Plugging-in the values of μ'_i and μ'_j in (5), we obtain:

$$\begin{aligned}\sigma'_{ij,2} &= \tfrac{1}{n+m}\textstyle\sum_{k=n+1}^{n+m}(p_{k,i}-\tfrac{n}{n+m}\mu_i-\tfrac{m}{n+m}\mu_i^m)(p_{k,j}-\tfrac{n}{n+m}\mu_j-\tfrac{m}{n+m}\mu_j^m) \\ &= \tfrac{1}{n+m}\textstyle\sum_{k=n+1}^{n+m}(p_{k,i}-\mu_i^m+\tfrac{n}{n+m}\mu_i^m-\tfrac{n}{n+m}\mu_i)(p_{k,j}-\mu_j^m+\tfrac{n}{n+m}\mu_j^m-\tfrac{n}{n+m}\mu_j) \\ &= \tfrac{1}{n+m}\textstyle\sum_{k=n+1}^{n+m}(p_{k,i}-\mu_i^m)(p_{k,j}-\mu_j^m) + \tfrac{1}{n+m}\sum_{k=n+1}^{n+m}(p_{k,i}-\mu_i^m)\tfrac{n}{n+m}(\mu_j^m-\mu_j)+ \\ &\quad \tfrac{1}{n+m}\textstyle\sum_{k=n+1}^{n+m}\tfrac{n}{n+m}(\mu_i^m-\mu_i)(p_{k,j}-\mu_j^m) + \tfrac{1}{n+m}\sum_{k=n+1}^{n+m}\tfrac{n}{n+m}(\mu_i^m-\mu_i)\tfrac{n}{n+m}(\mu_j^m-\mu_j).\end{aligned}$$

Since $\sum_{k=n+1}^{n+m}(p_{k,i} - \mu_i^m) = 0$, $1 \le i \le d$, we have

$$\sigma'_{ij,2} = \tfrac{m}{n+m}\sigma_{ij}^m + \tfrac{n^2m}{(n+m)^3}(\mu_i - \mu_i^m)(\mu_j - \mu_j^m), \tag{7}$$

where

$$\sigma_{ij}^m = \frac{1}{m}\sum_{k=n+1}^{n+m}(p_{k,i} - \mu_i^m)(p_{k,j} - \mu_j^m)), \quad 1 \le i, j \le d,$$

is the i, j-th element of the covariance matrix Σ_m of the point set P_m.

Finally, we have

$$\sigma'_{ij} = \sigma'_{ij,1} + \sigma'_{ij,2} = \frac{1}{n+m}(n\sigma_{ij} + m\sigma_{ij}^m) + \frac{nm}{(n+m)^2}(\mu_i - \mu_i^m)(\mu_j - \mu_j^m). \tag{8}$$

Note that σ_{ij}^m, and therefore σ'_{ij}, can be computed in $O(m)$ time. Thus, for a fixed dimension d, the covariance matrix Σ also can be computed in $O(m)$ time.

Deleting points

Let $P_m = \{\vec{p}_{n-m+1}, \vec{p}_{n-m}, \dots, \vec{p}_n\}$ be a subset of the point set P, and let $\vec{\mu}^m = (\mu_1^m, \mu_2^m, \dots, \mu_d^m)$ be the center of gravity of P_m. We subtract P_m from P, obtaining new point set P'. The j-th component, μ'_j, $1 \le j \le d$, of the center of gravity $\vec{\mu}' = (\mu'_1, \mu'_2, \dots, \mu'_d)$ of P' is

$$\begin{aligned} \mu'_j &= \tfrac{1}{n-m}\textstyle\sum_{k=1}^{n-m} p_{k,j} \\ &= \tfrac{1}{n-m}\left(\textstyle\sum_{k=1}^{n} p_{k,j} - \sum_{k=n-m+1}^{n} p_{k,j}\right) \\ &= \tfrac{n}{n-m}\mu_j - \tfrac{m}{n-m}\mu_j^m. \end{aligned}$$

The (i, j)-th component, σ'_{ij}, $1 \le i, j \le d$, of the covariance matrix Σ' of P' is

$$\begin{aligned} \sigma'_{ij} &= \tfrac{1}{n-m}\textstyle\sum_{k=1}^{n-m}(p_{k,i} - \mu'_i)(p_{k,j} - \mu'_j) \\ &= \tfrac{1}{n-m}\textstyle\sum_{k=1}^{n}(p_{k,i} - \mu'_i)(p_{k,j} - \mu'_j) - \tfrac{1}{n-m}\sum_{k=n-m+1}^{n}(p_{k,i} - \mu'_i)(p_{k,j} - \mu'_j). \end{aligned}$$

Let

$$\sigma'_{ij} = \sigma'_{ij,1} - \sigma'_{ij,2},$$

where,

$$\sigma'_{ij,1} = \frac{1}{n-m}\sum_{k=1}^{n}(p_{k,i} - \mu'_i)(p_{k,j} - \mu'_j), \tag{9}$$

and

$$\sigma'_{ij,2} = \frac{1}{n-m}\sum_{k=n-m+1}^{n}(p_{k,i} - \mu'_i)(p_{k,j} - \mu'_j). \tag{10}$$

Plugging-in the values of μ'_i and μ'_j in (9), we obtain:

$$\begin{aligned}
\sigma'_{ij,1} &= \tfrac{1}{n-m}\textstyle\sum_{k=1}^{n}(p_{k,i} - \tfrac{n}{n-m}\mu_i + \tfrac{m}{n-m}\mu_i^m)(p_{k,j} - \tfrac{n}{n-m}\mu_j + \tfrac{m}{n-m}\mu_j^m) \\
&= \tfrac{1}{n-m}\textstyle\sum_{k=1}^{m}(p_{k,i} - \mu_i + \tfrac{m}{n-m}\mu_i - \tfrac{m}{n-m}\mu_i^m)(p_{k,j} - \mu_j + \tfrac{m}{n-m}\mu_j - \tfrac{m}{n-m}\mu_j^m) \\
&= \tfrac{1}{n-m}\textstyle\sum_{k=1}^{n}(p_{k,i} - \mu_i)(p_{k,j} - \mu_j) + \tfrac{1}{n-m}\textstyle\sum_{k=1}^{n}(p_{k,i} - \mu_i)(\tfrac{m}{n-m}\mu_j - \tfrac{m}{n-m}\mu_j^m) + \\
&\quad \tfrac{1}{n-m}\textstyle\sum_{k=1}^{n}(\tfrac{m}{n-m}\mu_i - \tfrac{m}{n-m}\mu_i^m)(p_{k,j} - \mu_j) + \\
&\quad \tfrac{1}{n-m}\textstyle\sum_{k=1}^{n}(\tfrac{m}{n-m}\mu_i - \tfrac{m}{n-m}\mu_i^m)(\tfrac{m}{n-m}\mu_j - \tfrac{m}{n-m}\mu_i^m).
\end{aligned}$$

Since $\sum_{k=1}^{n}(p_{k,i} - \mu_i) = 0$, $1 \le i \le d$, we have

$$\sigma'_{ij,1} = \tfrac{n}{n-m}\sigma_{ij} + \tfrac{nm^2}{(n-m)^3}(\mu_i - \mu_i^m)(\mu_j - \mu_j^m). \qquad (11)$$

Plugging-in the values of μ'_i and μ'_j in (10), we obtain:

$$\begin{aligned}
\sigma'_{ij,2} &= \tfrac{1}{n-m}\textstyle\sum_{k=n-m+1}^{n}(p_{k,i} - \tfrac{n}{n-m}\mu_i + \tfrac{m}{n-m}\mu_i^m)(p_{k,j} - \tfrac{n}{n-m}\mu_j + \tfrac{m}{n-m}\mu_j^m) \\
&= \tfrac{1}{n-m}\textstyle\sum_{k=n-m+1}^{n}(p_{k,i} - \mu_i^m + \tfrac{n}{n-m}\mu_i^m - \tfrac{n}{n-m}\mu_i)(p_{k,j} - \mu_j^m + \tfrac{n}{n-m}\mu_j^m - \tfrac{n}{n-m}\mu_j) \\
&= \tfrac{1}{n-m}\textstyle\sum_{k=n-m+1}^{n}(p_{k,i} - \mu_i^m)(p_{k,j} - \mu_j^m) + \tfrac{1}{n-m}\textstyle\sum_{k=n-m+1}^{n-m}(p_{k,i} - \mu_i^m)\tfrac{n}{n-m}(\mu_j^m - \mu_j) + \\
&\quad \tfrac{1}{n-m}\textstyle\sum_{k=n-m+1}^{n}\tfrac{n}{n-m}(\mu_i^m - \mu_i)(p_{k,j} - \mu_j^m) + \tfrac{1}{n-m}\textstyle\sum_{k=n-m+1}^{n}\tfrac{n}{n-m}(\mu_i^m - \mu_i)\tfrac{n}{n-m}(\mu_j^m - \mu_j).
\end{aligned}$$

Since $\sum_{k=n-m+1}^{n}(p_{k,i} - \mu_i^m) = 0$, $1 \le i \le d$, we have

$$\sigma'_{ij,2} = \tfrac{n}{n-m}\sigma_{ij}^m + \tfrac{n^2m}{(n-m)^3}(\mu_i - \mu_i^m)(\mu_j - \mu_j^m), \qquad (12)$$

where

$$\sigma_{ij}^m = \frac{1}{m}\sum_{k=n-m+1}^{n}(p_{k,i} - \mu_i^m)(p_{k,j} - \mu_j^m)), \quad 1 \le i,j \le d,$$

is the i,j-th element of the covariance matrix Σ_m of the point set P_m.

Finally, we have

$$\sigma'_{ij} = \sigma'_{ij,1} + \sigma'_{ij,2} = \frac{1}{n-m}(n\sigma_{ij} - m\sigma_{ij}^m) - \frac{nm}{(n-m)^2}(\mu_i - \mu_i^m)(\mu_j - \mu_j^m). \qquad (13)$$

Note that σ_{ij}^m, and therefore σ'_{ij}, can be computed in $O(m)$ time. Thus, for a fixed dimension d, the covariance matrix Σ also can be computed in $O(m)$ time. □

As a corollary of (8), in the case when only one point, $\vec{p}_{n+1}$, is added to a point set P, the elements of the new covariance matrix are given by

$$\sigma'_{ij} = \sigma'_{ij,1} + \sigma'_{ij,2} = \frac{n}{n+1}\sigma_{ij} + \frac{n}{(n+1)^2}(p_{n+1,i} - \mu_i)(p_{n+1,j} - \mu_j), \qquad (14)$$

and can be computed in $O(1)$ time.

Similarly, as a corollary of (13), in the case when only one point, $\vec{p}_e$, is deleted from a point set P, the elements of the new covariance matrix are given by

$$\sigma'_{ij} = \sigma'_{ij,1} - \sigma'_{ij,2} = \frac{m}{m-1}\sigma_{ij} - \frac{m}{(m-1)^2}(p_{e,i} - \mu_i)(p_{e,j} - \mu_j), \qquad (15)$$

and also can be computed in $O(1)$ time.

The principal components of discrete point sets can be strongly influenced by point clusters [12]. To avoid the influence of the distribution of the point set, often continuous sets, especially the convex hull of a point set is considered, which lead to so-called continuous PCA. Computing PCA bounding boxes [14], [11], or retrieval of 3D-objects [23], are typical applications where continuous PCA are of interest. Due to better readability and compactness of the paper, we present the closed-form solutions for dynamic version of continuous PCA in the appendix. There, we consider cases when the point set is a convex polytope or the boundary of a convex polytope in $\mathbb{R}^2$ and $\mathbb{R}^3$.

3 An application - computing PCA bounding boxes

PCA is a well-established technique for dimensionality reduction and multivariate analysis, with numerous applications in both static and dynamic context. Examples of many applications of PCA include data compression, exploratory data analysis, visualization, image processing, pattern and image recognition, time series prediction, detecting perfect and reflective symmetry, and dimension detection (see textbooks [13] and [15] for thorough overview over PCA's applications).

In the rest of this section, we consider solutions for the static and dynamic versions of the bounding box problem.

3.1 Computing bounding boxes - static version

Substituting sets of points or complex geometric shapes with their bounding boxes is motivated by many applications. For example, in computer graphics, it is used to maintain hierarchical data structures for fast rendering of a scene or for collision detection. Additional applications include those in shape analysis and shape simplification, or in statistics, for storing and performing range-search queries on a large database of samples.

Computing a minimum-area bounding rectangle of a set of n points in $\mathbb{R}^2$ can be done in $O(n \log n)$ time, for example with the rotating calipers algorithm [22]. O'Rourke [19] presented a deterministic algorithm, a rotating calipers variant in $\mathbb{R}^3$, for computing the minimum-volume bounding box of a set of n points in $\mathbb{R}^3$. His algorithm requires $O(n^3)$ time and $O(n)$ space. Barequet and Har-Peled [4] have contributed two algorithms with nearly linear complexity, based on a core-set approach, that compute $(1+\varepsilon)$-approximations of the minimum-volume bounding box of point sets in $\mathbb{R}^3$. The running times of their algorithms are $O(n + 1/\varepsilon^{4.5})$ and $O(n \log n + n/\varepsilon^3)$, respectively. A further improvement to $O(n + 1/\varepsilon^3)$ running time can be obtained by using a coreset of size $O(1/\varepsilon)$ by Agarwal, Har-Peled, and Varadarajan [2], and Chan [8].

Numerous heuristics have been proposed for computing a box that encloses a given set of points. The simplest heuristic is naturally to compute the axis-aligned bounding box of the point set. Two-dimensional variants of this heuristic include the well-known *R-tree*, the *packed R-tree* [20], the R^*-*tree* [5], the R^+-*tree* [21], etc.

A frequently used heuristic for computing a bounding box of a set of points is based on PCA. The principal components of the point set define the axes of the bounding box. Once the directions of the axes are given, the dimension of the bounding box is easily found by the extreme values of the projection of the points on the corresponding axis.

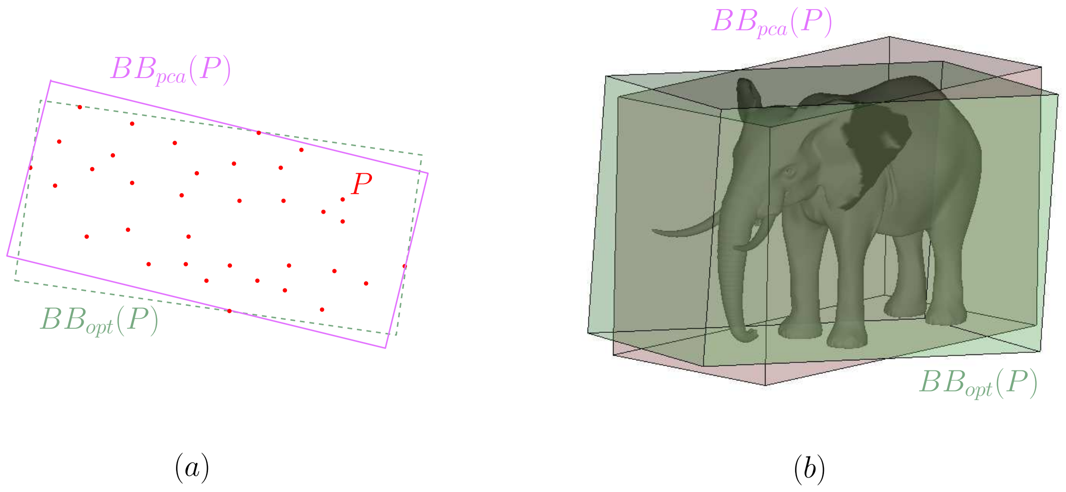


Figure 1: The minimum-area(volume) bounding box and the PCA bounding box of a point set P, (a) in $\mathbb{R}^2$, and (b) in $\mathbb{R}^3$.

Two distinguished applications of this heuristic are the OBB-tree [14] and the BOX-TREE [3], hierarchical bounding box structures, that support efficient collision detection and ray tracing. Computing a bounding box of a set of points in $\mathbb{R}^2$ and $\mathbb{R}^3$ by PCA is simple and requires linear time. To avoid the influence of the distribution of the point set on the directions of the PCs, a possible approach is to consider the convex hull, or the boundary of the convex hull $CH(P)$ of the point set P. Thus, the complexity of the algorithm increases to $O(n \log n)$. The popularity of this heuristic, besides its speed, lies in its easy implementation and in the fact that usually PCA bounding boxes are tight-fitting, see Figure 1 for an illustration. Experimental results of the quality of the PCA bounding boxes can be found in [11],[16], and theoretical results in [12].

3.2 Computing bounding boxes - dynamic version

Dynamic $(1+\varepsilon)$-approximation bounding box can be efficiently solved by the dynamic data structure [9] that can maintain an ε-coreset of n points in $O(\log n)$ time for any constant $\varepsilon > 0$ and any constant dimension. That is an improvement of the previous method by Agarwal, Har-Peled, and Varadarajan [2] that requires polylogarithmic update time. However, both results are more of theoretical importance, since there realization involves sophisticated data structures difficult for implementation.

Computing PCA bounding boxes of a point set consists of two steps: 1) computing the principal components, that define the axes of the bounding box, and 2) computing the extremal point along the axes, that determine the size of the bounding box. In Section 2, we have presented closed-form solution for efficient update of the principal components when we add or delete several points from the point set. In sequel, we consider step 2.

3.2.1 Computing extremal points

A trivial way to compute the extremal points is to scan all points, which takes linear time.

Faster algorithms that compute extremal points dynamically are related to computing and maintaining the convex hull of the point set dynamically. Then, one can perform extreme point queries in polylogarithmic time. Here, we give an overview of the known results of dynamical computation of convex hull and some of related operation on it in $\mathbb{R}^2$ and $\mathbb{R}^3$.

Brodal and Jacob [6] present a data structure that maintains a finite set of n points in the plane under point insertions and point deletions in amortized $O(\log n)$ time per operation. This data structure requires $O(n)$ space, and supports extreme point queries in a given direction, tangent queries through a given point, and queries for the neighboring points on the convex hull in $O(\log n)$ time. T. Chan [7] presents a fully dynamic randomized data structure that can answer queries about the convex hull of a set of n points in three dimensions, where insertions take $O(\log^3 n)$ expected amortized time, deletions take $O(\log^6 n)$ expected amortized time, and extreme-point queries take $O(\log^2 n)$ worst-case time. This is the first method that guarantees polylogarithmic update and query cost for arbitrary sequences of insertions and deletions, and improves the previous $O(n^\varepsilon)$-time method by Agarwal and Matoušek [1].

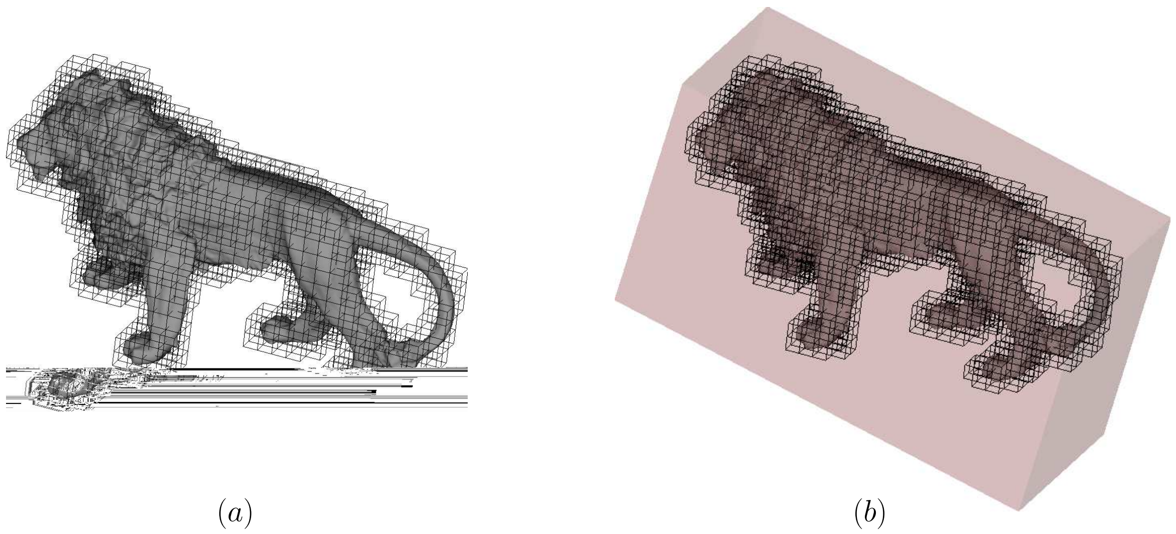
There are two disadvantages in the above approaches. First, they require a computation of the convex hull, which increases the complexity of the algorithms to $O(n \log n)$. However, the convex hull computation and building corresponding data structures can be done in preprocessing, which is not critical for many applications. Second, the above date structures for dynamic convex hull computation are of theoretical importance, they are quite difficult for implementation, and to best of our knowledge, they have never been implemented. Therefore, in the next section, we consider two simple approaches for computing extremal points, one is the already mentioned linear scan of all points, and the other is a grid approach, refined with several variants.

4 Practical variants of dynamical PCA bounding boxes and experimental results

The main focus in this section is to show the advantages of the theoretical results presented in this paper in the context of computing dynamic PCA bounding boxes. We present three practical simple algorithms, and compare their performances. A thorough comparison study of different variants of statical PCA bounding boxes the interested reader could find in [11]. The algorithms were implemented in C#, C++ and OpenGL, and tested on a Core Duo 2.33GHz with 2GB memory. The principal components of all algorithms are computed with the closed-form solutions from Section 2. They differ only how the extremal points along the principal components are found. The implemented algorithms are the following:

- **PCA-AP** (PCA-all-points) - finds the extremal points by going through all points.

- **PCA-AGP** (PCA-all-grid-points) -the space is discretized by a regular three dimensional axis-aligned grid, with a cube of size $\varepsilon \times \varepsilon \times \varepsilon$ as primer component. See Figure 2 for an illustration. The grid size is chosen relatively to the size of the object. Each object is scaled such that its diameter is 1. The values of ε are between 0.001 and 1. The corners of non-empty cells are considered to find the extremal points along the principal directions.



(a) (b)

Figure 2: (a) A real world object and its corresponding grid for $\varepsilon = 0.03$. Only the non-empty grids are visualized. (b) The bounding box of the object obtained by the PCA-AGP algorithm.

- **PCA-EGP** (PCA-extremal-grid-points) - this is an improvement of the PCA-AGP algorithm. To each vertical grid line, i.e., orthogonal to XY plane, two extremal corners of the non-empty cell are computed. Thus, we reduced the candidates for extremal points from $O(\frac{1}{\varepsilon^3})$ to $O(\frac{1}{\varepsilon^2})$.

We further reduce the number of points considered in the PCA-AGP and PCA-EGP algorithms by replacing the cell corners with the centers of gravity of the cells. Afterwords, we expand the resulting box by $\sqrt{3}\varepsilon/2$ to ensure that the box contains all original points. We have implemented also these variants, but, since for a reasonable big grid size ($\varepsilon \geq 0.01$) the running time improvements are negligible, we report here only the results of the base variants of the algorithms PCA-AGP and PCA-EGP. However, for very dense grid the improved version of the both algorithms give better results.

In the following experiments, we add (delete) random points from the point set, and compare the results of a dynamical versions of PCA bounding boxes with the their corresponding statical versions (when the covariance matrix of the point set is computed from scratch). The time of computing, the volume of a bounding box, and the grid density are parameters of interest in this evaluation study. The test were performed on big number of real graphics models taken from various publicly available sources (Stanford 3D scanning repository, 3D Cafe). Typical samples of the results are given in Table 1, Table 2, and Table 3.

Table 1: Time needed by the PCA bounding box algorithms for the lion model (183408 points). The values in the table are the average of results of 100 runs of the algorithms, each time adding/deleting the corresponding number of points.

Adding/deleting points, $\varepsilon = 0.005$						
	1pnt	1pnt	100 pnts	100 pnts	1000 pnts	1000 pnts
algorithm	static	dynamic	static	dynamic	static	dynamic
PCA-AP	0.166 s	0.014 s	0.171 s	0.015 s	0.172 s	0.016 s
PCA-AGP	0.092 s	0.0095 s	0.093 s	0.0085 s	0.99 s	0.017 s
PCA-EGP	0.0805 s	0.0055 s	0.082 s	0.006 s	0.092 s	0.0135 s

The main conclusions of the experiments are as follows:

- As expected from the theoretical results, the dynamic versions of the algorithms are significantly faster than their static counterparts. Typically, the dynamic versions are about an oder of magnitude faster (see Table 1).

- The dynamic PCA-AP algorithm is not only significantly faster than its statical version, it is also faster than the static version of the PCA-AGP and PCA-EGP algorithms. This is due to fact that the brute force manner of finding the extremal points is faster than computing the covariance matrix of the new point set from scratch, although both algorithms require $O(n)$ time in the asymptotic analysis.

- Clearly, the PCA-AGP and PCA-EGP algorithms, that exploit the grid subdivision structure, are faster than the PCA-AP algorithm. The price that must be paid for this is two-folded. First, an extra preprocessing time for building the grid is needed. For the example considered in Table 1, computing the grid takes about 0.4 seconds for the PCA-AGP algorithm, and about 0.43 for the PCA-EGP algorithm. Second, the resulting bounding boxes are less precise (see Table 2).

- As it is shown in Table 3, for grids that are not very sparse ($\varepsilon \leq 0.03$), the approximated PCA bounding boxes computed by the PCA-AGP and PCA-EGP algorithms are quite close to the exact PCA bounding boxes.

Tight bounding boxes for the PCA-AGP and PCA-EGP algorithms can be obtained by the following approach. Let P_1 be the supporting plane at the extremal grid point along one principal direction, and let P_2 be the plane parallel to P_1, such that the distance

Table 2: Volume of the PCA bounding box algorithms for the lion model. The values in the table are the average of results of 100 runs of the algorithms, each time adding the corresponding number of points.

Adding points, dynamic version, $\varepsilon = 0.005$					
algorithm	1pnt	10pnt	100 pnts	1000 pnts	10000 pnts
PCA-AP	285.5	644.6	856.3	1149.1	1236.4
PCA-AGP, PCA-EGP	295.5	662.7	880.3	1221.8	1263.2

Table 3: Volumes of the PCA bounding boxes algorithms for lion model for different grid density. The values in the table are the average of results of 100 runs of the algorithms, each time adding the corresponding number of points.

Adding 100 points, dynamic version						
algorithm	$\varepsilon = 0.005$	$\varepsilon = 0.01$	$\varepsilon = 0.03$	$\varepsilon = 0.05$	$\varepsilon = 0.1$	$\varepsilon = 0.2$
PCA-AP	856.3	856.3	856.3	856.3	856.3	856.3
PCA-AGP, PCA-EGP	880.3	904.3	942.3	1080.1	1292.7	2324.8

between P_1 and P_2 is $\sqrt{3}\varepsilon/2$, and P_2 intersect or is tangent to the grid. We denote by S the subspace between P_1 and P_2. Then, the candidates points for the chosen principal direction, that determine the tight bounding box, are all original points that belong to cells that have intersection with S. See Fig. 3 for an illustration. However, in the worst case all original points have to be checked.

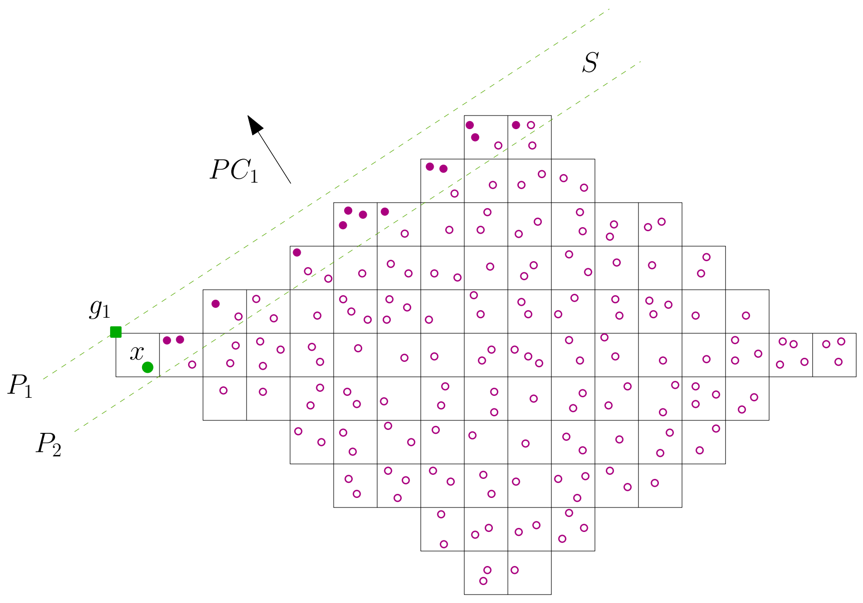


Figure 3: For the principal direction PC_1, the algorithms PCA-AGP and PCA-EGP detect the point g_1 as extremal grid point, and the point x as extremal point of the original point set. However, there are other points (the violet colored circles) that are further than x along PC_1.

Further (theoretical) improvement of the algorithms presented here could be obtained if, instead of the point set, we consider its convex hull when we look for extremal points. This only makes sense if the convex hull is computed dynamically. Otherwise, computing the static convex hull of the points will be more expensive than finding the exact extremal points by scanning all points.

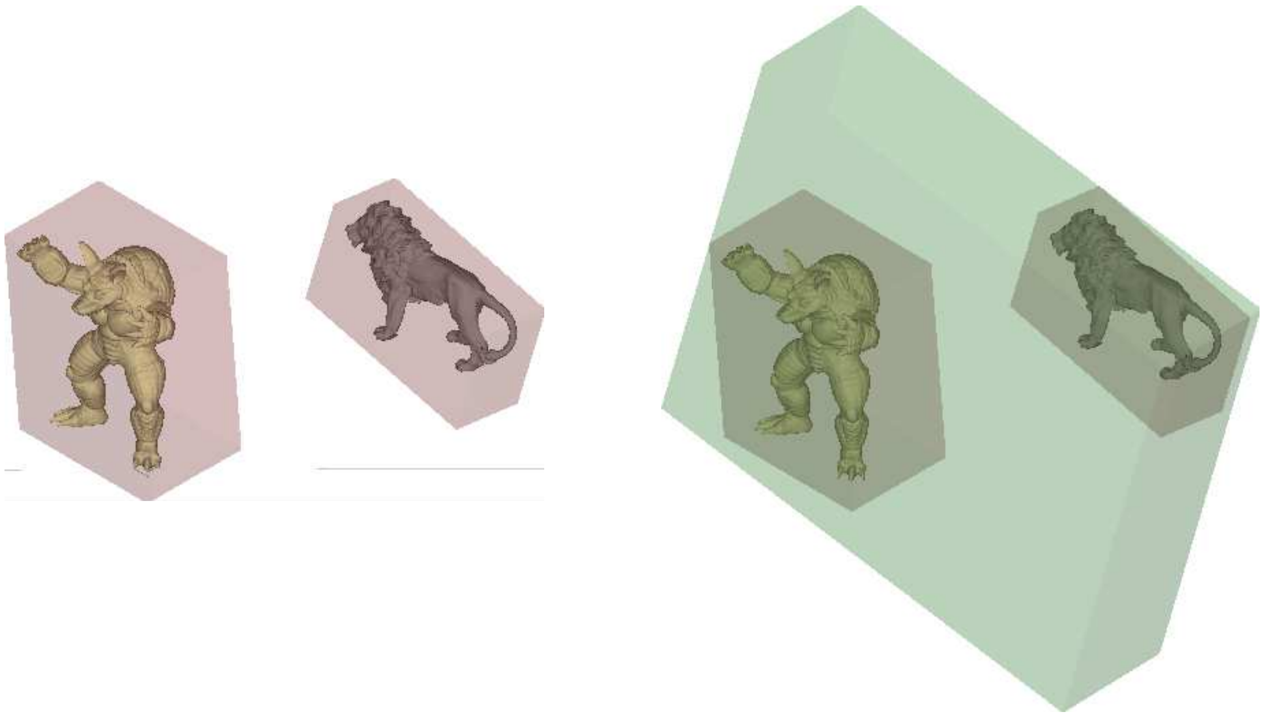


Figure 4: Left: two objects with their PCA bounding boxes. Right: the common PCA bounding box. Computing the common PCA bounding box dynamically takes 0.004 seconds, while the static version takes 0.02 seconds.

4.0.2 Computing efficiently a bounding box of several objects

An interesting application of the closed-form solutions from Section 2 is to compute the principal components of two or more objects with already known covariance matrices. Since σ_{ij} and σ_{ij}^m in (8) and (15) are previously known, σ'_{ij} can be computed in $O(1)$ time. Thus, for fixed d the new covariance matrix Σ and the new principal components can be computed also in $O(1)$ time. This is a significant improvement over the commonly used approach to compute the principal components from scratch, which take time linear in the number of points. Efficient computation of the common PCA bounding box of several object is straightforward. See Fig. 4 for an illustration in $\mathbb{R}^3$.

5 Conclusion and future work

The main contribution of this paper are the closed-form solutions for updating the principal components of a dynamic point set. The new principal components can be computed in constant time, when a constant number of points are added or deleted from the point set. This is a significant improvement of the commonly used approach, when the new principal components are computed from scratch, which takes linear time. The advantages of the theoretical results were verified and presented in the context of computing dynamic PCA bounding boxes, a very important application in many fields including computer graphics, where the PCA boxes are used to maintain hierarchical data structures for fast rendering of a scene or for collision detection. We have presented three practical simple algorithms and compare their performances.

In the appendix we consider the computation of the principal components of a dynamic continuous point set. We give closed form-solutions when the point set is a convex polytope or the boundary of a convex polytope in $\mathbb{R}^2$ or $\mathbb{R}^3$.

An interesting open problem is to find a closed-form solution for dynamical point sets different from convex polyhedra, for example, implicit surfaces or B-splines. An implementation of computing principal components in a dynamic and continuous setting is planned for future work. Applications of the results presented here in other fields, like computer vision or visualization, are of high interest.

There are several further improvements and open problems regarding computing dynamic PCA bounding boxes. Instead of subdividing the space by a simple regular grid, one can use more sophisticated data structures, like octrees or binary space partition-trees to speed up the time needed to find the extremal points along the principal directions. A practical, implementable algorithm for computing the dynamic convex hull of the point set (computing extremal point dynamically) would also improve the dynamic PCA bounding box algorithms. Finding coresets for dynamic PCA bounding boxes will lead to efficient approximation algorithms for PCA bounding boxes. We are also not aware of data structures for efficient computation of extremal points both approximately and dynamically. Such data structures are also of interest.

References

[1] P. K. Agarwal, J. Matoušek. *Dynamic half-space range reporting and its applications.* Algorithmica, (**13**) (1995) 325–345.

[2] P. K. Agarwal, S. Har-Peled, K.R. Varadarajan. *Approximating extent measures of points.* J. ACM, (**51**) (2004) 606–635.

[3] G. Barequet, B. Chazelle, L. J. Guibas, J. S. B. Mitchell, and A. Tal. *Boxtree: A hierarchical representation for surfaces in 3D.* Computer Graphics Forum, (**15**) (1996) 387–396.

[4] G. Barequet, and S. Har-Peled. *Efficiently approximating the minimum-volume bounding box of a point set in three dimensions.* Journal of Algorithms, **38**(**1**) (2001) 91–109.

[5] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. *The R^*-tree: An efficient and robust access method for points and rectangles.* In Proceedings of ACM SIGMOD International Conference on Management of Data, pages 322–331, 1990.

[6] G. S. Brodal, R. Jacob. *Dynamic planar convex hull.* In Proceedings of the 43rd IEEE Sympos. Found. Comput. Sci., (2002) 617–626.

[7] T. M. Chan. *A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries.* In Proceedings of SODA '06, pages 1196–1202, 2006.

[8] T. M. Chan. *Faster core-set constructions and data-stream algorithms in fixed dimensions.* Computational Geometry: Theory and Applications, (**35**) (2006) 20–35.

[9] T. M. Chan. *Dynamic coresets.* Discrete and Computational Geometry, (**42**) (2009) 469–488.

[10] S.-W. Cheng, Y. Wang, and Z. Wu. *Provable dimension detection dsing principal component analysis.* Int. J. Comput. Geometry Appl., (**18**) (2008) 415–440.

[11] D. Dimitrov, M. Holst, C. Knauer, K. Kriegel. *Closed-form solutions for continuous PCA and bounding box algorithms.* A. Ranchordas et al. (Eds.): VISIGRAPP 2008, CCIS, Springer, (**24**) (2009) 26-40.

[12] D. Dimitrov, C. Knauer, K. Kriegel, G. Rote. *Bounds on the quality of the PCA bounding boxes.* Computational Geometry, (**42**) (2009) 772–789.

[13] R. Duda, P. Hart, D. Stork. *Pattern classification.* John Wiley & Sons, Inc., 2nd ed., 2001.

[14] S. Gottschalk, M. C. Lin, and D. Manocha. *OBBTree: A hierarchical structure for rapid interference detection.* Computer Graphics, (**30**) (1996) 171–180.

[15] I. Jolliffe. *Principal component analysis.* Springer-Verlag, New York, 2nd ed., 2002.

[16] M. Lahanas, T. Kemmerer, N. Milickovic, D. B. K. Karouzakis, and N. Zamboglou. *Optimized bounding boxes for three-dimensional treatment planning in brachytherapy.* Medical Physics, (**27**) (2000) 2333–2342.

[17] B. N. Parlett. *The symmetric eigenvalue problem.* Society of Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998.

[18] W. H. Press, S. A. Teukolsky, W. T. Veterling, and B. P. Flannery. *Numerical recipes in C: the art of scientific computing.* Cambridge University Press, New York, USA, 2nd ed., 1995.

[19] J. O'Rourke. *Finding minimal enclosing boxes.* International Journal of Computer and Information Science, (**14**) (1985) 183–199.

[20] N. Roussopoulos, and D. Leifker. *Direct spatial search on pictorial databases using packed R-trees.* In ACM SIGMOD, pages 17–31, 1985.

[21] T. Sellis, N. Roussopoulos, and C. Faloutsos. *The R^+-tree: A dynamic index for multidimensional objects.* In Proceedings of the 13th Very Large Data Bases (VLDB) Conference, pages 507–518, 1987.

[22] G. Toussaint. *Solving geometric problems with the rotating calipers.* In Proceedings of the 2nd IEEE Mediterranean Electrotechnical Conference (MELECON), May 1983.

[23] D. V. Vranić, D. Saupe, and J. Richter. *Tools for 3D-object retrieval: Karhunen-Loeve transform and spherical harmonics.* In Proceeding of the IEEE 2001 Workshop Multimedia Signal Processing, pages 293–298, 2001.

6 Appendix

Updating the principal components efficiently - continuous case

Here, we consider the computation of the principal components of a dynamic continuous point set. We present a closed form-solutions when the point set is a convex polytope or a boundary of a convex polytope in $\mathbb{R}^2$ or $\mathbb{R}^3$. When the point set is a boundary of a convex polytope, we can update the new principal components in $O(k)$ time, for both deletion and addition, under the assumption that we know the k facets in which the polytope changes. Under the same assumption, when the point set is a convex polytope in $\mathbb{R}^2$ or $\mathbb{R}^3$, we can update the principal components in $O(k)$ time after adding points. But, to update the principal components after deleting points from a convex polytope in $\mathbb{R}^2$ or $\mathbb{R}^3$ we need $O(n)$ time. This is due to the fact that, after a deletion the center of gravity of the old convex hull (polyhedron) could lie outside the new convex hull, and therefore, a retetrahedralization is needed (see Subsection 6.1.1 and Subsection 6.2.1 for details).

6.1 Continuous PCA in $\mathbb{R}^3$

6.1.1 Continuous PCA over a (convex) polyhedron in $\mathbb{R}^3$

Let P be a point set in $\mathbb{R}^3$, and let X be its convex hull. We assume that the boundary of X is triangulated (if it is not, we can triangulate it in preprocessing). We choose an arbitrary point $\vec{o}$ in the interior of X, for example, we can choose that $\vec{o}$ is the center of gravity of the boundary of X. Each triangle from the boundary together with $\vec{o}$ forms a tetrahedron. Let the number of such formed tetrahedra be n. The k-th tetrahedron, with vertices $\vec{x}_{1,k}, \vec{x}_{2,k}, \vec{x}_{3,k}, \vec{x}_{4,k} = \vec{o}$, can be represented in a parametric form by $\vec{Q}_i(s,t,u) = \vec{x}_{4,i} + s\,(\vec{x}_{1,i} - \vec{x}_{4,i}) + t\,(\vec{x}_{2,i} - \vec{x}_{4,i}) + u\,(\vec{x}_{3,i} - \vec{x}_{4,i})$, for $0 \le s,t,u \le 1$, and $s+t+u \le 1$. For $1 \le i \le 3$, we use $x_{i,j,k}$ to denote the i-th coordinate of the vertex $\vec{x}_j$ of the polyhedron $\vec{Q}_k$.

The center of gravity of the k-th tetrahedron is

$$\vec{\mu}_k \;=\; \frac{\int_0^1 \int_0^{1-s} \int_0^{1-s-t} \rho(\vec{Q}_k(s,t))\vec{Q}_i(s,t)\,du\,dt\,ds}{\int_0^1 \int_0^{1-s} \int_0^{1-s-t} \rho(\vec{Q}_k(s,t))\,du\,dt\,ds},$$

where $\rho(\vec{Q}_k(s,t))$ is a mass density at a point $\vec{Q}_k(s,t)$. Since, we can assume $\rho(\vec{Q}_k(s,t)) = 1$, we have

$$\vec{\mu}_k = \frac{\int_0^1 \int_0^{1-s} \int_0^{1-s-t} \vec{Q}_k(s,t)\, du\, dt\, ds}{\int_0^1 \int_0^{1-s} \int_0^{1-s-t} du\, dt\, ds} = \frac{\vec{x}_{1,k}+\vec{x}_{2,k}+\vec{x}_{3,k}+\vec{x}_{4,k}}{4}.$$

The contribution of each tetrahedron to the center of gravity of X is proportional to its volume. If M_k is the 3×3 matrix whose l-th row is $\vec{x}_{l,k} - \vec{x}_{4,k}$, for $l = 1 \ldots 3$, then the volume of the k-th tetrahedron is

$$v_k = \text{volume}(Q_k) = \frac{|det(M_k)|}{3!}.$$

We introduce a weight to each tetrahedron that is proportional with its volume, define as

$$w_k = \frac{v_k}{\sum_{k=1}^n v_k} = \frac{v_k}{v},$$

where v is the volume of X. Then, the center of gravity of X is

$$\vec{\mu} = \sum_{k=1}^n w_k \vec{\mu}_k.$$

The covariance matrix of the k-th tetrahedron is

$$\begin{aligned}\Sigma_k &= \frac{\int_0^1 \int_0^{1-s} \int_0^{1-s-t} (\vec{Q}_k(s,t,u)-\vec{\mu})\,(\vec{Q}_k(s,t,u)-\vec{\mu})^T\, du\, dt\, ds}{\int_0^1 \int_0^{1-s} \int_0^{1-s-t} du\, dt\, ds} \\ &= \frac{1}{20}\Big(\sum_{j=1}^4 \sum_{h=1}^4 (\vec{x}_{j,k} - \vec{\mu})(\vec{x}_{h,k} - \vec{\mu})^T + \\ &\qquad \sum_{j=1}^4 (\vec{x}_{j,k} - \vec{\mu})(\vec{x}_{j,k} - \vec{\mu})^T \Big).\end{aligned}$$

The (i,j)-th element of Σ_k, $i,j \in \{1,2,3\}$, is

$$\begin{aligned}\sigma_{ij,k} &= \frac{1}{20}\Big(\sum_{l=1}^4 \sum_{h=1}^4 (x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\ &\qquad \sum_{l=1}^4 (x_{i,l,k} - \mu_i)(x_{j,l,k} - \mu_j) \Big),\end{aligned}$$

with $\vec{\mu} = (\mu_1, \mu_2, \mu_3)$. Finally, the covariance matrix of X is

$$\Sigma = \sum_{i=1}^n w_i \Sigma_i,$$

with (i,j)-th element

$$\begin{aligned}\sigma_{ij} &= \frac{1}{20}\Big(\sum_{k=1}^n \sum_{l=1}^4 \sum_{h=1}^4 w_i (x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\ &\qquad \sum_{k=1}^n \sum_{l=1}^4 w_i (x_{i,l,k} - \mu_i)(x_{j,l,k} - \mu_j) \Big).\end{aligned}$$

We would like to note that the above expressions hold also for any non-convex polyhedron that can be tetrahedralized. A star-shaped object, where $\vec{o}$ is the kernel of the object, is such example.

Adding points

We add points to P, obtaining a new point set P'. Let X' be the convex hull of P'. We consider that X' is obtained from X by deleting n_d, and adding n_a tetrahedra. Let

$$v' = \sum_{k=1}^{n} v_k + \sum_{k=1}^{n_a} v_k - \sum_{k=1}^{n_d} v_k = v + \sum_{k=1}^{n_a} v_k - \sum_{k=1}^{n_d} v_k.$$

The center of gravity of X' is

$$\begin{aligned} \vec{\mu}' &= \textstyle\sum_{k=1}^{n} w'_k \vec{\mu}_k + \sum_{k=1}^{n_a} w'_k \vec{\mu}_k - \sum_{k=1}^{n_d} w'_k \vec{\mu}_k \\ &= \tfrac{1}{v'} \left(\textstyle\sum_{k=1}^{n} v_k \vec{\mu}_k + \sum_{k=1}^{n_a} v_k \vec{\mu}_k - \sum_{k=1}^{n_d} v_k \vec{\mu}_k \right) \\ &= \tfrac{1}{v'} \left(v\vec{\mu} + \textstyle\sum_{k=1}^{n_a} v_k \vec{\mu}_k - \sum_{k=1}^{n_d} v_k \vec{\mu}_k \right). \end{aligned} \tag{16}$$

Let

$$\vec{\mu}_a = \frac{1}{v'} \sum_{k=1}^{n_a} v_k \vec{\mu}_k, \quad \text{and} \quad \vec{\mu}_d = \frac{1}{v'} \sum_{k=1}^{n_d} v_k \vec{\mu}_k.$$

Then, we can rewrite (16) as

$$\vec{\mu}' = \frac{v}{v'} \vec{\mu} + \vec{\mu}_a - \vec{\mu}_d. \tag{17}$$

The i-th component of $\vec{\mu}_a$ and $\vec{\mu}_d$, $1 \le i \le 3$, is denoted by $\mu_{i,a}$ and $\mu_{i,d}$, respectively. The (i,j)-th component, σ'_{ij}, $1 \le i,j \le 3$, of the covariance matrix Σ' of X' is

$$\begin{aligned} \sigma'_{ij} = \ & \tfrac{1}{20} \Big(\textstyle\sum_{k=1}^{n} \sum_{l=1}^{4} \sum_{h=1}^{4} w'_k (x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j) + \\ & \textstyle\sum_{k=1}^{n} \sum_{l=1}^{4} w'_k (x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j) \Big) + \\ & \tfrac{1}{20} \Big(\textstyle\sum_{k=1}^{n_a} \sum_{l=1}^{4} \sum_{h=1}^{4} w'_k (x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j) + \\ & \textstyle\sum_{k=1}^{n_a} \sum_{l=1}^{4} w'_k (x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j) - \\ & \textstyle\sum_{k=1}^{n_d} \sum_{l=1}^{4} \sum_{h=1}^{4} w'_k (x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j) - \\ & \textstyle\sum_{k=1}^{n_d} \sum_{l=1}^{4} w'_k (x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j) \Big). \end{aligned}$$

Let

$$\sigma'_{ij} = \frac{1}{20} (\sigma'_{ij,11} + \sigma'_{ij,12} + \sigma'_{ij,21} + \sigma'_{ij,22} - \sigma'_{ij,31} - \sigma'_{ij,32}),$$

where,

$$\sigma'_{ij,11} = \sum_{k=1}^{n} \sum_{l=1}^{4} \sum_{h=1}^{4} w'_k (x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j), \tag{18}$$

$$\sigma'_{ij,12} = \sum_{k=1}^{n}\sum_{l=1}^{4} w'_k(x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j), \tag{19}$$

$$\sigma'_{ij,21} = \sum_{k=1}^{n_a}\sum_{l=1}^{4}\sum_{h=1}^{4} w'_k(x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j), \tag{20}$$

$$\sigma'_{ij,22} = \sum_{k=1}^{n_a}\sum_{l=1}^{4} w'_k(x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j), \tag{21}$$

$$\sigma'_{ij,31} = \sum_{k=1}^{n_d}\sum_{l=1}^{4}\sum_{h=1}^{4} w'_k(x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j), \tag{22}$$

$$\sigma'_{ij,32} = \sum_{k=1}^{n_d}\sum_{l=1}^{4} w'_k(x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j). \tag{23}$$

Plugging-in the values of μ'_i and μ'_j in (18), we obtain:

$$\begin{aligned}
\sigma'_{ij,11} &= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{4}\sum_{h=1}^{4} w'_k(x_{i,l,k} - \frac{v}{v'}\mu_i - \mu_{i,a} + \mu_{i,d})(x_{j,h,k} - \frac{v}{v'}\mu_j - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{4}\sum_{h=1}^{4} w'_k(x_{i,l,k} - \mu_i + \mu_i(1 - \frac{v}{v'}) - \mu_{i,a} + \mu_{i,d}) \\
&\qquad\qquad (x_{j,h,k} - \mu_j + \mu_j(1 - \frac{v}{v'}) - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{4}\sum_{h=1}^{4} w'_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{4}\sum_{h=1}^{4} w'_k(x_{i,l,k} - \mu_i)(\mu_j(1 - \frac{v}{v'}) - \mu_{j,a} + \mu_{j,d}) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{4}\sum_{h=1}^{4} w'_k(\mu_i(1 - \frac{v}{v'}) - \mu_{i,a} + \mu_{i,d})(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{4}\sum_{h=1}^{4} w'_k(\mu_i(1 - \frac{v}{v'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \frac{v}{v'}) - \mu_{j,a} + \mu_{j,d}).
\end{aligned} \tag{24}$$

Since $\sum_{k=1}^{n}\sum_{l=1}^{4} w'_k(x_{i,l,k} - \mu_i) = 0$, $1 \le i \le 3$, we have

$$\begin{aligned}
\sigma'_{ij,11} &= \textstyle\frac{1}{v'}\sum_{k=1}^{n}\sum_{l=1}^{4}\sum_{h=1}^{4} v_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\frac{1}{v'}\sum_{k=1}^{n}\sum_{l=1}^{4}\sum_{h=1}^{4} v_k(\mu_i(1 - \frac{v}{v'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \frac{v}{v'}) - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\frac{1}{v'}\sum_{k=1}^{n}\sum_{l=1}^{4}\sum_{h=1}^{4} v_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad 16\tfrac{v}{v'}(\mu_i(1 - \tfrac{v}{v'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \tfrac{v}{v'}) - \mu_{j,a} + \mu_{j,d}).
\end{aligned} \tag{25}$$

Plugging-in the values of μ'_i and μ'_j in (19), we obtain:

$$
\begin{aligned}
\sigma'_{ij,12} &= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{4} w'_k(x_{i,l,k} - \frac{v}{v'}\mu_i - \mu_{i,a} + \mu_{i,d})(x_{j,h,k} - \frac{v}{v'}\mu_j - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{4} w'_k(x_{i,l,k} - \mu_i + \mu_i(1-\frac{v}{v'}) - \mu_{i,a} + \mu_{i,d}) \\
&\qquad\qquad (x_{j,h,k} - \mu_j + \mu_j(1-\frac{v}{v'}) - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{4} w'_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{4} w'_k(x_{i,l,k} - \mu_i)(\mu_j(1-\frac{v}{v'}) - \mu_{j,a} + \mu_{j,d}) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{4} w'_k(\mu_i(1-\frac{v}{v'}) - \mu_{i,a} + \mu_{i,d})(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{4} w'_k(\mu_i(1-\frac{v}{v'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1-\frac{v}{v'}) - \mu_{j,a} + \mu_{j,d}).
\end{aligned} \tag{26}
$$

Since $\sum_{k=1}^{n}\sum_{l=1}^{4} w'_k(x_{i,l,k} - \mu_i) = 0$, $1 \le i \le 3$, we have

$$
\begin{aligned}
\sigma'_{ij,12} &= \textstyle\frac{1}{v'}\sum_{k=1}^{n}\sum_{l=1}^{4} v_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\frac{1}{v'}\sum_{k=1}^{n}\sum_{l=1}^{4} v_k(\mu_i(1-\frac{v}{v'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1-\frac{v}{v'}) - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\frac{1}{v'}\sum_{k=1}^{n}\sum_{l=1}^{4} v_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle 4\frac{v}{v'}(\mu_i(1-\frac{v}{v'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1-\frac{v}{v'}) - \mu_{j,a} + \mu_{j,d}).
\end{aligned} \tag{27}
$$

From (26) and (27), we obtain

$$
\begin{aligned}
\sigma'_{ij,1} &= \sigma'_{ij,11} + \sigma'_{ij,12} \\
&= \sigma_{ij} + 20\tfrac{v}{v'}(\mu_i(1-\tfrac{v}{v'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1-\tfrac{v}{v'}) - \mu_{j,a} + \mu_{j,d}).
\end{aligned} \tag{28}
$$

Note that $\sigma'_{ij,1}$ can be computed in $O(1)$ time. The components $\sigma'_{ij,21}$ and $\sigma'_{ij,22}$ can be computed in $O(n_a)$ time, while $O(n_d)$ time is needed for computing $\sigma'_{ij,31}$ and $\sigma'_{ij,32}$. Thus, $\vec{\mu'}$ and

$$
\begin{aligned}
\sigma'_{ij} &= \tfrac{1}{20}(\sigma'_{ij,11} + \sigma'_{ij,12} + \sigma'_{ij,21} + \sigma'_{ij,22} + \sigma'_{ij,31} + \sigma'_{ij,32}) \\
&= \tfrac{1}{20}(\sigma_{ij} + \sigma'_{ij,21} + \sigma'_{ij,22} + \sigma'_{ij,31} + \sigma_{ij,32}) + \\
&\quad \tfrac{v}{v'}(\mu_i(1-\tfrac{v}{v'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1-\tfrac{v}{v'}) - \mu_{j,a} + \mu_{j,d})
\end{aligned} \tag{29}
$$

can be computed in $O(n_a + n_d)$ time.

Deleting points

Let the new convex hull be obtained by deleting n_d tetrahedra from and added n_a tetrahedra to the old convex hull. If the interior point $\vec{o}$ (needed for a tetrahedronization of a convex polytope), after deleting points, lies inside the new convex hull, then the same formulas and time complexity, as by adding points, follow. If $\vec{o}$ lie outside the new convex hull, then, we need to choose a new interior point $\vec{o}'$, and recompute the new tetrahedra associated with it. Thus, we need in total $O(n)$ time to update the principal components.

Under certain assumptions, we can recompute the new principal components faster:

- If we know that a certain point of the polyhedron will never be deleted, we can choose $\vec{o}$ to be that point. In that case, we also have the same closed-formed solution as for adding a point.
- Let the facets of the convex polyhedron have similar (uniformly distributed) area. We choose $\vec{o}$ to be the center of gravity of the polyhedron. Then, we can expect that after deleting a point, $\vec{o}$ will remain in the new convex hull. However, after several deletion, $\vec{o}$ could lie outside the convex hull, and then we need to recompute it and the associate tetrahedra with it.

Note, that in the case when we consider boundary of a convex polyhedron (Subsection 6.1.2 and Subsection 6.2.2), we do not need an interior point $\vec{o}$ and the same time complexity holds for both adding and deleting points.

6.1.2 Continuous PCA over a boundary of a polyhedron

Let X be a polyhedron in $\mathbb{R}^3$. We assume that the boundary of X is triangulated (if it is not, we can triangulate it in preprocessing), containing n triangles. The k-th triangle, with vertices $\vec{x}_{1,k}, \vec{x}_{2,k}, \vec{x}_{3,k}$, can be represented in a parametric form by $\vec{T}_k(s,t) = \vec{x}_{1,k} + s\,(\vec{x}_{2,k} - \vec{x}_{1,k}) + t\,(\vec{x}_{3,k} - \vec{x}_{1,k})$, for $0 \le s, t \le 1$, and $s + t \le 1$. For $1 \le i \le 3$, we denote by $x_{i,j,k}$ the i-th coordinate of the vertex $\vec{x}_j$ of the triangle $\vec{T}_k$.

The center of gravity of the k-th triangle is

$$\vec{\mu}_k = \frac{\int_0^1 \int_0^{1-s} \vec{T}_i(s,t)\,dt\,ds}{\int_0^1 \int_0^{1-s} dt\,ds} = \frac{\vec{x}_{1,k} + \vec{x}_{2,k} + \vec{x}_{3,k}}{3}.$$

The contribution of each triangle to the center of gravity of the triangulated surface is proportional to its area. The area of the k-th triangle is

$$a_k = \text{area}(T_k) = \frac{|(\vec{x}_{2,k} - \vec{x}_{1,k})| \times |(\vec{x}_{3,k} - \vec{x}_{1,k})|}{2}.$$

We introduce a weight to each triangle that is proportional with its area, define as

$$w_k = \frac{a_k}{\sum_{i=1}^n a_k} = \frac{a_k}{a},$$

where a is the area of X. Then, the center of gravity of the boundary of X is

$$\vec{\mu} = \sum_{k=1}^n w_k \vec{\mu}_k.$$

The covariance matrix of the k-th triangle is

$$\begin{aligned} \Sigma_k &= \frac{\int_0^1 \int_0^{1-s} (\vec{T}_k(s,t) - \vec{\mu})\,(\vec{T}_k(s,t) - \vec{\mu})^T\,dt\,ds}{\int_0^1 \int_0^{1-s} dt\,ds} \\ &= \frac{1}{12}\Big(\sum_{j=1}^3 \sum_{h=1}^3 (\vec{x}_{j,k} - \vec{\mu})(\vec{x}_{h,k} - \vec{\mu})^T + \\ &\qquad \sum_{j=1}^3 (\vec{x}_{j,k} - \vec{\mu})(\vec{x}_{j,k} - \vec{\mu})^T \Big). \end{aligned}$$

The (i,j)-th element of Σ_k, $i,j \in \{1,2,3\}$, is

$$\begin{aligned}\sigma_{ij,k} &= \tfrac{1}{12}\Big(\textstyle\sum_{l=1}^{3}\sum_{h=1}^{3}(x_{i,l,k}-\mu_i)(x_{j,h,k}-\mu_j)+ \\ &\qquad \textstyle\sum_{l=1}^{3}(x_{i,l,k}-\mu_i)(x_{j,l,k}-\mu_j)\Big),\end{aligned}$$

with $\vec{\mu} = (\mu_1, \mu_2, \mu_3)$. Finally, the covariance matrix of the boundary of X is

$$\Sigma = \textstyle\sum_{k=1}^{n} w_k \Sigma_k.$$

Adding points

We add points to X. Let X' be the new convex hull. We assume that X' is obtained from X by deleting n_d, and adding n_a tetrahedra. Then the sum of the areas of all triangles is

$$a' = \sum_{k=1}^{n} a_k + \sum_{k=1}^{n_a} a_k - \sum_{k=1}^{n_d} a_k = a + \sum_{k=1}^{n_a} a_k - \sum_{k=1}^{n_d} a_k.$$

The center of gravity of X' is

$$\begin{aligned}\vec{\mu}' &= \textstyle\sum_{k=1}^{n} w_k'\vec{\mu}_k + \sum_{k=1}^{n_a} w_k'\vec{\mu}_k - \sum_{k=1}^{n_d} w_k'\vec{\mu}_k \\ &= \tfrac{1}{a'}\left(\textstyle\sum_{k=1}^{n} a_k\vec{\mu}_k + \sum_{k=1}^{n_a} a_k\vec{\mu}_k - \sum_{k=1}^{n_d} a_k\vec{\mu}_k\right) \\ &= \tfrac{1}{a'}\left(a\vec{\mu} + \textstyle\sum_{k=1}^{n_a} a_k\vec{\mu}_k - \sum_{k=1}^{n_d} a_k\vec{\mu}_k\right).\end{aligned} \tag{30}$$

Let

$$\vec{\mu}_a = \frac{1}{a'}\sum_{k=1}^{n_a} a_k\vec{\mu}_k, \quad \text{and} \quad \vec{\mu}_d = \frac{1}{a'}\sum_{k=1}^{n_d} a_k\vec{\mu}_k.$$

Then, we can rewrite (30) as

$$\vec{\mu}' = \frac{a}{a'}\vec{\mu} + \vec{\mu}_a - \vec{\mu}_d. \tag{31}$$

The i-th component of $\vec{\mu}_a$ and $\vec{\mu}_d$, $1 \le i \le 3$, is denoted by $\mu_{i,a}$ and $\mu_{i,d}$, respectively. The (i,j)-th component, σ'_{ij}, $1 \le i,j \le 3$, of the covariance matrix Σ' of X' is

$$\begin{aligned}\sigma'_{ij} &= \tfrac{1}{12}\Big(\textstyle\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} w_k'(x_{i,l,k}-\mu_i')(x_{j,h,k}-\mu_j')+ \\ &\qquad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3} w_k'(x_{i,l,k}-\mu_i')(x_{j,l,k}-\mu_j')\Big)+ \\ &\quad \tfrac{1}{12}\Big(\textstyle\sum_{k=1}^{n_a}\sum_{l=1}^{3}\sum_{h=1}^{3} w_k'(x_{i,l,k}-\mu_i')(x_{j,h,k}-\mu_j')+ \\ &\qquad \textstyle\sum_{k=1}^{n_a}\sum_{l=1}^{3} w_k'(x_{i,l,k}-\mu_i')(x_{j,l,k}-\mu_j')- \\ &\qquad \textstyle\sum_{k=1}^{n_d}\sum_{l=1}^{3}\sum_{h=1}^{3} w_k'(x_{i,l,k}-\mu_i')(x_{j,h,k}-\mu_j')- \\ &\qquad \textstyle\sum_{k=1}^{n_d}\sum_{l=1}^{3} w_k'(x_{i,l,k}-\mu_i')(x_{j,l,k}-\mu_j')\Big).\end{aligned}$$

Let

$$\sigma'_{ij} = \frac{1}{12}(\sigma'_{ij,11} + \sigma'_{ij,12} + \sigma'_{ij,21} + \sigma'_{ij,22} - \sigma'_{ij,31} - \sigma'_{ij,32}),$$

where,

$$\sigma'_{ij,11} = \sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} w'_k(x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j), \tag{32}$$

$$\sigma'_{ij,12} = \sum_{k=1}^{n}\sum_{l=1}^{3} w'_k(x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j), \tag{33}$$

$$\sigma'_{ij,21} = \sum_{k=1}^{n_a}\sum_{l=1}^{3}\sum_{h=1}^{3} w'_k(x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j), \tag{34}$$

$$\sigma'_{ij,22} = \sum_{k=1}^{n_a}\sum_{l=1}^{3} w'_k(x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j), \tag{35}$$

$$\sigma'_{ij,31} = \sum_{k=1}^{n_d}\sum_{l=1}^{3}\sum_{h=1}^{3} w'_k(x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j), \tag{36}$$

$$\sigma'_{ij,32} = \sum_{k=1}^{n_d}\sum_{l=1}^{3} w'_k(x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j). \tag{37}$$

Plugging-in the values of μ'_i and μ'_j in (32), we obtain:

$$\begin{aligned}
\sigma'_{ij,11} &= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} w'_k(x_{i,l,k} - \frac{a}{a'}\mu_i - \mu_{i,a} + \mu_{i,d})(x_{j,h,k} - \frac{a}{a'}\mu_j - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} w'_k(x_{i,l,k} - \mu_i + \mu_i(1 - \frac{a}{a'}) - \mu_{i,a} + \mu_{i,d}) \\
&\qquad\qquad (x_{j,h,k} - \mu_j + \mu_j(1 - \frac{a}{a'}) - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} w'_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} w'_k(x_{i,l,k} - \mu_i)(\mu_j(1 - \frac{a}{a'}) - \mu_{j,a} + \mu_{j,d}) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} w'_k(\mu_i(1 - \frac{a}{a'}) - \mu_{i,a} + \mu_{i,d})(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} w'_k(\mu_i(1 - \frac{a}{a'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \frac{a}{a'}) - \mu_{j,a} + \mu_{j,d}).
\end{aligned} \tag{38}$$

Since $\sum_{k=1}^{n}\sum_{l=1}^{3} w'_k(x_{i,l,k} - \mu_i) = 0$, $1 \leq i \leq 3$, we have

$$\begin{aligned}\sigma'_{ij,11} &= \tfrac{1}{a'}\textstyle\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} a_k(x_{i,l,k}-\mu_i)(x_{j,h,k}-\mu_j)+\\ &\quad \tfrac{1}{a'}\textstyle\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} a_k(\mu_i(1-\tfrac{a}{a'})-\mu_{i,a}+\mu_{i,d})(\mu_j(1-\tfrac{a}{a'})-\mu_{j,a}+\mu_{j,d})\\ &= \tfrac{1}{a'}\textstyle\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} a_k(x_{i,l,k}-\mu_i)(x_{j,h,k}-\mu_j)+\\ &\quad 9\tfrac{a}{a'}(\mu_i(1-\tfrac{a}{a'})-\mu_{i,a}+\mu_{i,d})(\mu_j(1-\tfrac{a}{a'})-\mu_{j,a}+\mu_{j,d}).\end{aligned} \tag{39}$$

Plugging-in the values of μ'_i and μ'_j in (33), we obtain:

$$\begin{aligned}\sigma'_{ij,12} &= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3} w'_k(x_{i,l,k}-\tfrac{a}{a'}\mu_i-\mu_{i,a}+\mu_{i,d})(x_{j,h,k}-\tfrac{a}{a'}\mu_j-\mu_{j,a}+\mu_{j,d})\\ &= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3} w'_k(x_{i,l,k}-\mu_i+\mu_i(1-\tfrac{a}{a'})-\mu_{i,a}+\mu_{i,d})\\ &\qquad\qquad (x_{j,h,k}-\mu_j+\mu_j(1-\tfrac{a}{a'})-\mu_{j,a}+\mu_{j,d})\\ &= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3} w'_k(x_{i,l,k}-\mu_i)(x_{j,h,k}-\mu_j)+\\ &\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3} w'_k(x_{i,l,k}-\mu_i)(\mu_j(1-\tfrac{a}{a'})-\mu_{j,a}+\mu_{j,d})+\\ &\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3} w'_k(\mu_i(1-\tfrac{a}{a'})-\mu_{i,a}+\mu_{i,d})(x_{j,h,k}-\mu_j)+\\ &\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3} w'_k(\mu_i(1-\tfrac{a}{a'})-\mu_{i,a}+\mu_{i,d})(\mu_j(1-\tfrac{a}{a'})-\mu_{j,a}+\mu_{j,d}).\end{aligned} \tag{40}$$

Since $\sum_{k=1}^{n}\sum_{l=1}^{3} w'_k(x_{i,l,k}-\mu_i)=0$, $1\le i\le 3$, we have

$$\begin{aligned}\sigma'_{ij,12} &= \tfrac{1}{a'}\textstyle\sum_{k=1}^{n}\sum_{l=1}^{3} a_k(x_{i,l,k}-\mu_i)(x_{j,h,k}-\mu_j)+\\ &\quad \tfrac{1}{a'}\textstyle\sum_{k=1}^{n}\sum_{l=1}^{3} a_k(\mu_i(1-\tfrac{a}{a'})-\mu_{i,a}+\mu_{i,d})(\mu_j(1-\tfrac{a}{a'})-\mu_{j,a}+\mu_{j,d})\\ &= \tfrac{1}{a'}\textstyle\sum_{k=1}^{n}\sum_{l=1}^{3} a_k(x_{i,l,k}-\mu_i)(x_{j,h,k}-\mu_j)+\\ &\quad 3\tfrac{a}{a'}(\mu_i(1-\tfrac{a}{a'})-\mu_{i,a}+\mu_{i,d})(\mu_j(1-\tfrac{a}{a'})-\mu_{j,a}+\mu_{j,d}).\end{aligned} \tag{41}$$

From (40) and (41), we obtain

$$\begin{aligned}\sigma'_{ij,1} &= \sigma'_{ij,11}+\sigma'_{ij,12}\\ &= \sigma_{ij}+12\tfrac{a}{a'}(\mu_i(1-\tfrac{a}{a'})-\mu_{i,a}+\mu_{i,d})(\mu_j(1-\tfrac{a}{a'})-\mu_{j,a}+\mu_{j,d}).\end{aligned} \tag{42}$$

Note that $\sigma'_{ij,1}$ can be computed in $O(1)$ time. The components $\sigma'_{ij,21}$ and $\sigma'_{ij,22}$ can be computed in $O(n_a)$ time, while $O(n_d)$ time is needed for computing $\sigma'_{ij,31}$ and $\sigma'_{ij,32}$. Thus, $\vec{\mu}'$ and

$$\begin{aligned}\sigma'_{ij} &= \tfrac{1}{12}(\sigma'_{ij,11}+\sigma'_{ij,12}+\sigma'_{ij,21}+\sigma'_{ij,22}+\sigma'_{ij,31}+\sigma'_{ij,32})\\ &= \tfrac{1}{12}(\sigma_{ij}+\sigma'_{ij,21}+\sigma'_{ij,22}+\sigma'_{ij,31}+\sigma_{ij,32})+\\ &\quad \tfrac{a}{a'}(\mu_i(1-\tfrac{a}{a'})-\mu_{i,a}+\mu_{i,d})(\mu_j(1-\tfrac{a}{a'})-\mu_{j,a}+\mu_{j,d}).\end{aligned} \tag{43}$$

can be computed in $O(n_a + n_d)$ time.

Deleting points

Let the new convex hull be obtained by deleting n_d tetrahedra from and added n_a tetrahedra to the old convex hull. Consequently, the same formulas and time complexity, as by adding points, follow.

6.2 Continuous PCA in $\mathbb{R}^2$

6.2.1 Continuous PCA over a polygon

We assume that the polygon X is triangulated (if it is not, we can triangulate it in preprocessing), and the number of triangles is n. The k-th triangle, with vertices $\vec{x}_{1,k}, \vec{x}_{2,k}, \vec{x}_{3,k} = \vec{o}$, can be represented in a parametric form by $\vec{T}_i(s,t) = \vec{x}_{3,k} + s\,(\vec{x}_{1,k} - \vec{x}_{3,k}) + t\,(\vec{x}_{2,k} - \vec{x}_{3,k})$, for $0 \le s, t \le 1$, and $s + t \le 1$.

The center of gravity of the k-th triangle is

$$\vec{\mu}_i = \frac{\int_0^1 \int_0^{1-s} \vec{T}_i(s,t)\, dt\, ds}{\int_0^1 \int_0^{1-s} dt\, ds} = \frac{\vec{x}_{1,k} + \vec{x}_{2,k} + \vec{x}_{3,k}}{3}.$$

The contribution of each triangle to the center of gravity of X is proportional to its area. The area of the i-th triangle is

$$a_k = \text{area}(T_k) = \frac{|(\vec{x}_{2,k} - \vec{x}_{1,k})| \times |(\vec{x}_{3,k} - \vec{x}_{1,k})|}{2},$$

where $\times$ denotes the vector product. We introduce a weight to each triangle that is proportional with its area, define as

$$w_k = \frac{a_k}{\sum_{k=1}^n a_k} = \frac{a_k}{a},$$

where a is the area of X.Then, the center of gravity of X is

$$\vec{\mu} = \sum_{k=1}^{n} w_k \vec{\mu}_k.$$

The covariance matrix of the k-th triangle is

$$\begin{aligned}\Sigma_k &= \frac{\int_0^1 \int_0^{1-s} (\vec{T}_k(s,t) - \vec{\mu})\,(\vec{T}_k(s,t) - \vec{\mu})^T\, dt\, ds}{\int_0^1 \int_0^{1-s} dt\, ds} \\ &= \frac{1}{12}\Big(\sum_{j=1}^3 \sum_{h=1}^3 (\vec{x}_{j,k} - \vec{\mu})(\vec{x}_{h,k} - \vec{\mu})^T + \\ &\qquad \sum_{j=1}^3 (\vec{x}_{j,k} - \vec{\mu})(\vec{x}_{j,k} - \vec{\mu})^T \Big).\end{aligned}$$

The (i,j)-th element of Σ_k, $i, j \in \{1,2\}$, is

$$\begin{aligned}\sigma_{ij,k} &= \frac{1}{12}\Big(\sum_{l=1}^3 \sum_{h=1}^3 (x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\ &\qquad \sum_{l=1}^3 (x_{i,l,k} - \mu_i)(x_{j,l,k} - \mu_j) \Big),\end{aligned}$$

with $\vec{\mu} = (\mu_1, \mu_2)$. The covariance matrix of X is

$$\Sigma \;=\; \textstyle\sum_{k=1}^{n} w_k \Sigma_k.$$

Adding points

We add points to X. Let X' be the new convex hull. We assume that X' is obtained from X by deleting n_d, and adding n_a triangles. Then the sum of the areas of all triangles is

$$a' = \sum_{k=1}^{n} a_k + \sum_{k=1}^{n_a} a_k - \sum_{k=1}^{n_d} a_k = a + \sum_{k=1}^{n_a} a_k - \sum_{k=1}^{n_d} a_k.$$

The center of gravity of X' is

$$\begin{aligned}
\vec{\mu}' &= \textstyle\sum_{k=1}^{n} w'_k \vec{\mu}_k + \sum_{k=1}^{n_a} w'_k \vec{\mu}_k - \sum_{k=1}^{n_d} w'_k \vec{\mu}_k \\
&= \tfrac{1}{a'} \left(\textstyle\sum_{k=1}^{n} a_k \vec{\mu}_k + \sum_{k=1}^{n_a} a_k \vec{\mu}_k - \sum_{k=1}^{n_d} a_k \vec{\mu}_k \right) \qquad (44) \\
&= \tfrac{1}{a'} \left(a\vec{\mu} + \textstyle\sum_{k=1}^{n_a} a_k \vec{\mu}_k - \sum_{k=1}^{n_d} a_k \vec{\mu}_k \right).
\end{aligned}$$

Let

$$\vec{\mu}_a = \frac{1}{a'} \sum_{k=1}^{n_a} a_k \vec{\mu}_k, \quad \text{and} \quad \vec{\mu}_d = \frac{1}{a'} \sum_{k=1}^{n_d} a_k \vec{\mu}_k.$$

Then, we can rewrite (44) as

$$\vec{\mu}' = \frac{a}{a'} \vec{\mu} + \vec{\mu}_a - \vec{\mu}_d. \qquad (45)$$

The i-th component of $\vec{\mu}_a$ and $\vec{\mu}_d$, $1 \le i \le 2$, is denoted by $\mu_{i,a}$ and $\mu_{i,d}$, respectively. The (i,j)-th component, σ'_{ij}, $1 \le i,j \le 2$, of the covariance matrix Σ' of X' is

$$\begin{aligned}
\sigma'_{ij} \;=\; & \tfrac{1}{12}\Big(\textstyle\sum_{k=1}^{n} \sum_{l=1}^{3} \sum_{h=1}^{3} w'_k (x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j) + \\
& \textstyle\sum_{k=1}^{n} \sum_{l=1}^{3} w'_k (x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j) \Big) + \\
& \tfrac{1}{12}\Big(\textstyle\sum_{k=1}^{n_a} \sum_{l=1}^{3} \sum_{h=1}^{3} w'_k (x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j) + \\
& \textstyle\sum_{k=1}^{n_a} \sum_{l=1}^{3} w'_k (x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j) - \\
& \textstyle\sum_{k=1}^{n_d} \sum_{l=1}^{3} \sum_{h=1}^{3} w'_k (x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j) - \\
& \textstyle\sum_{k=1}^{n_d} \sum_{l=1}^{3} w'_k (x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j) \Big).
\end{aligned}$$

Let

$$\sigma'_{ij} = \frac{1}{12}(\sigma'_{ij,11} + \sigma'_{ij,12} + \sigma'_{ij,21} + \sigma'_{ij,22} - \sigma'_{ij,31} - \sigma'_{ij,32}),$$

where,

$$\sigma'_{ij,11} = \sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} w'_k(x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j), \tag{46}$$

$$\sigma'_{ij,12} = \sum_{k=1}^{n}\sum_{l=1}^{3} w'_k(x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j), \tag{47}$$

$$\sigma'_{ij,21} = \sum_{k=1}^{n_a}\sum_{l=1}^{3}\sum_{h=1}^{3} w'_k(x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j), \tag{48}$$

$$\sigma'_{ij,22} = \sum_{k=1}^{n_a}\sum_{l=1}^{3} w'_k(x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j), \tag{49}$$

$$\sigma'_{ij,31} = \sum_{k=1}^{n_d}\sum_{l=1}^{3}\sum_{h=1}^{3} w'_k(x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j), \tag{50}$$

$$\sigma'_{ij,32} = \sum_{k=1}^{n_d}\sum_{l=1}^{3} w'_k(x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j). \tag{51}$$

Plugging-in the values of μ'_i and μ'_j in (46), we obtain:

$$\begin{aligned}
\sigma'_{ij,11} &= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} w'_k(x_{i,l,k} - \frac{a}{a'}\mu_i - \mu_{i,a} + \mu_{i,d})(x_{j,h,k} - \frac{a}{a'}\mu_j - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} w'_k(x_{i,l,k} - \mu_i + \mu_i(1 - \frac{a}{a'}) - \mu_{i,a} + \mu_{i,d}) \\
&\qquad\qquad (x_{j,h,k} - \mu_j + \mu_j(1 - \frac{a}{a'}) - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} w'_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} w'_k(x_{i,l,k} - \mu_i)(\mu_j(1 - \frac{a}{a'}) - \mu_{j,a} + \mu_{j,d}) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} w'_k(\mu_i(1 - \frac{a}{a'}) - \mu_{i,a} + \mu_{i,d})(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} w'_k(\mu_i(1 - \frac{a}{a'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \frac{a}{a'}) - \mu_{j,a} + \mu_{j,d}).
\end{aligned} \tag{52}$$

Since $\sum_{k=1}^{n}\sum_{l=1}^{3} w'_k(x_{i,l,k} - \mu_i) = 0$, $1 \le i \le 2$, we have

$$\begin{aligned}
\sigma'_{ij,11} &= \textstyle\frac{1}{a'}\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} a_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\frac{1}{a'}\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} a_k(\mu_i(1 - \frac{a}{a'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \frac{a}{a'}) - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\frac{1}{a'}\sum_{k=1}^{n}\sum_{l=1}^{3}\sum_{h=1}^{3} a_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad 9\textstyle\frac{a}{a'}(\mu_i(1 - \frac{a}{a'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \frac{a}{a'}) - \mu_{j,a} + \mu_{j,d}).
\end{aligned} \tag{53}$$

Plugging-in the values of μ'_i and μ'_j in (47), we obtain:

$$\begin{aligned}
\sigma'_{ij,12} &= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3} w'_k(x_{i,l,k} - \frac{a}{a'}\mu_i - \mu_{i,a} + \mu_{i,d})(x_{j,h,k} - \frac{a}{a'}\mu_j - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3} w'_k(x_{i,l,k} - \mu_i + \mu_i(1 - \frac{a}{a'}) - \mu_{i,a} + \mu_{i,d}) \\
&\qquad\qquad (x_{j,h,k} - \mu_j + \mu_j(1 - \frac{a}{a'}) - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3} w'_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3} w'_k(x_{i,l,k} - \mu_i)(\mu_j(1 - \frac{a}{a'}) - \mu_{j,a} + \mu_{j,d}) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3} w'_k(\mu_i(1 - \frac{a}{a'}) - \mu_{i,a} + \mu_{i,d})(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{3} w'_k(\mu_i(1 - \frac{a}{a'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \frac{a}{a'}) - \mu_{j,a} + \mu_{j,d}).
\end{aligned} \tag{54}$$

Since $\sum_{k=1}^{n}\sum_{l=1}^{3} w'_k(x_{i,l,k} - \mu_i) = 0$, $1 \le i \le 2$, we have

$$\begin{aligned}
\sigma'_{ij,12} &= \textstyle\frac{1}{a'}\sum_{k=1}^{n}\sum_{l=1}^{3} a_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\frac{1}{a'}\sum_{k=1}^{n}\sum_{l=1}^{3} a_k(\mu_i(1 - \frac{a}{a'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \frac{a}{a'}) - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\frac{1}{a'}\sum_{k=1}^{n}\sum_{l=1}^{3} a_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad 3\textstyle\frac{a}{a'}(\mu_i(1 - \frac{a}{a'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \frac{a}{a'}) - \mu_{j,a} + \mu_{j,d}).
\end{aligned} \tag{55}$$

From (54) and (55), we obtain

$$\begin{aligned}
\sigma'_{ij,1} &= \sigma'_{ij,11} + \sigma'_{ij,12} \\
&= \sigma_{ij} + 12\textstyle\frac{a}{a'}(\mu_i(1 - \frac{a}{a'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \frac{a}{a'}) - \mu_{j,a} + \mu_{j,d}).
\end{aligned} \tag{56}$$

Note that $\sigma'_{ij,1}$ can be computed in $O(1)$ time. The components $\sigma'_{ij,21}$ and $\sigma'_{ij,22}$ can be computed in $O(n_a)$ time, while $O(n_d)$ time is needed for computing $\sigma'_{ij,31}$ and $\sigma'_{ij,32}$. Thus, $\vec{\mu'}$ and

$$\begin{aligned}
\sigma'_{ij} &= \textstyle\frac{1}{12}(\sigma'_{ij,11} + \sigma'_{ij,12} + \sigma'_{ij,21} + \sigma'_{ij,22} + \sigma'_{ij,31} + \sigma'_{ij,32}) \\
&= \textstyle\frac{1}{12}(\sigma_{ij} + \sigma'_{ij,21} + \sigma'_{ij,22} + \sigma'_{ij,31} + \sigma_{ij,32}) + \\
&\quad \textstyle\frac{a}{a'}(\mu_i(1 - \frac{a}{a'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \frac{a}{a'}) - \mu_{j,a} + \mu_{j,d}).
\end{aligned} \tag{57}$$

can be computed in $O(n_a + n_d)$ time.

Deleting points

Let the new convex hull be obtained by deleting n_d tetrahedra from and added n_a tetrahedra to the old convex hull. If the interior point $\vec{o}$, after deleting points, lies inside the new convex hull, then the same formulas and time complexity, as by adding points, follow. However, $\vec{o}$ could lie outside the new convex hull. Then, we need to choose a new interior point $\vec{o'}$, and recompute the new tetrahedra associated with it. Thus, we need in total $O(n)$ time to update the principal components.

6.2.2 Continuous PCA over the boundary of a polygon

Let X be a polygon in $\mathbb{R}^2$. We assume that the boundary of X is comprised of n line segments. The k-th line segment, with vertices $\vec{x}_{1,k}, \vec{x}_{2,k}$, can be represented in a parametric form by

$$\vec{S}_k(t) = \vec{x}_{1,k} + t\,(\vec{x}_{2,k} - \vec{x}_{1,k}).$$

Since we assume that the mass density is constant, the center of gravity of the k-th line segment is

$$\vec{\mu}_k = \frac{\int_0^1 \vec{S}_k(t)\,dt}{\int_0^1 dt} = \frac{\vec{x}_{1,k} + \vec{x}_{2,k}}{2}.$$

The contribution of each line segment to the center of gravity of the boundary of a polygon is proportional with the length of the line segment. The length of the k-th line segment is

$$s_k = \text{length}(S_k) = ||\vec{x}_{2,k} - \vec{x}_{1,k}||.$$

We introduce a weight to each line segment that is proportional with its length, define as

$$w_k = \frac{s_k}{\sum_{k=1}^n s_k} = \frac{s_k}{s},$$

where s is the perimeter of X. Then, the center of gravity of the boundary of X is

$$\vec{\mu} = \sum_{k=1}^{n} w_k \vec{\mu}_k.$$

The covariance matrix of the k-th line segment is

$$\begin{aligned} \Sigma_k &= \frac{\int_0^1 (\vec{S}_k(t) - \vec{\mu})\,(\vec{S}_k(t) - \vec{\mu})^T\,dt}{\int_0^1 dt} \\ &= \frac{1}{6}\Big(\sum_{j=1}^2 \sum_{h=1}^2 (\vec{x}_{j,k} - \vec{\mu})(\vec{x}_{h,k} - \vec{\mu})^T + \\ &\qquad \sum_{j=1}^2 (\vec{x}_{j,k} - \vec{\mu})(\vec{x}_{j,k} - \vec{\mu})^T \Big). \end{aligned}$$

The (i,j)-th element of Σ_k, $i,j \in \{1,2\}$, is

$$\begin{aligned} \sigma_{ij,k} &= \frac{1}{6}\Big(\sum_{l=1}^2 \sum_{h=1}^2 (x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\ &\qquad \sum_{l=1}^2 (x_{i,l,k} - \mu_i)(x_{j,l,k} - \mu_j) \Big), \end{aligned}$$

with $\vec{\mu} = (\mu_1, \mu_2)$.

The covariance matrix of the boundary of X is

$$\Sigma = \sum_{k=1}^n w_k \Sigma_k.$$

Adding points

We add points to X. Let X' be the new convex hull. We assume that X' is obtained from X by deleting n_d, and adding n_a line segments. Then the sum of the lengths of all line segments is

$$s' = \sum_{k=1}^{n} l_k + \sum_{k=1}^{n_a} s_k - \sum_{k=1}^{n_d} s_k = s + \sum_{k=1}^{n_a} s_k - \sum_{k=1}^{n_d} s_k.$$

The center of gravity of X' is

$$\begin{aligned} \vec{\mu}' &= \textstyle\sum_{k=1}^{n} w'_k \vec{\mu}_k + \sum_{k=1}^{n_a} w'_k \vec{\mu}_k - \sum_{k=1}^{n_d} w'_k \vec{\mu}_k \\ &= \tfrac{1}{s'} \left(\textstyle\sum_{k=1}^{n} s_k \vec{\mu}_k + \sum_{k=1}^{n_a} s_k \vec{\mu}_k - \sum_{k=1}^{n_d} s_k \vec{\mu}_k \right) \\ &= \tfrac{1}{s'} \left(s\vec{\mu} + \textstyle\sum_{k=1}^{n_a} s_k \vec{\mu}_k - \sum_{k=1}^{n_d} s_k \vec{\mu}_k \right). \end{aligned} \tag{58}$$

Let

$$\vec{\mu}_a = \frac{1}{s'} \sum_{k=1}^{n_a} s_k \vec{\mu}_k, \quad \text{and} \quad \vec{\mu}_d = \frac{1}{s'} \sum_{k=1}^{n_d} s_k \vec{\mu}_k.$$

Then, we can rewrite (58) as

$$\vec{\mu}' = \frac{s}{s'} \vec{\mu} + \vec{\mu}_a - \vec{\mu}_d. \tag{59}$$

The i-th component of $\vec{\mu}_a$ and $\vec{\mu}_d$, $1 \le i \le 2$, is denoted by $\mu_{i,a}$ and $\mu_{i,d}$, respectively. The (i,j)-th component, σ'_{ij}, $1 \le i,j \le 2$, of the covariance matrix Σ' of X' is

$$\begin{aligned} \sigma'_{ij} = {} & \tfrac{1}{6}\Big(\textstyle\sum_{k=1}^{n} \sum_{l=1}^{2} \sum_{h=1}^{2} w'_k (x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j) + \\ & \quad \textstyle\sum_{k=1}^{n} \sum_{l=1}^{2} w'_k (x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j) \Big) + \\ & \tfrac{1}{6}\Big(\textstyle\sum_{k=1}^{n_a} \sum_{l=1}^{2} \sum_{h=1}^{2} w'_k (x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j) + \\ & \quad \textstyle\sum_{k=1}^{n_a} \sum_{l=1}^{2} w'_k (x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j) - \\ & \quad \textstyle\sum_{k=1}^{n_d} \sum_{l=1}^{2} \sum_{h=1}^{2} w'_k (x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j) - \\ & \quad \textstyle\sum_{k=1}^{n_d} \sum_{l=1}^{2} w'_k (x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j) \Big). \end{aligned}$$

Let

$$\sigma'_{ij} = \frac{1}{6}(\sigma'_{ij,11} + \sigma'_{ij,12} + \sigma'_{ij,21} + \sigma'_{ij,22} - \sigma'_{ij,31} - \sigma'_{ij,32}),$$

where,

$$\sigma'_{ij,11} = \sum_{k=1}^{n} \sum_{l=1}^{2} \sum_{h=1}^{2} w'_k (x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j), \tag{60}$$

$$\sigma'_{ij,12} = \sum_{k=1}^{n}\sum_{l=1}^{2} w'_k(x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j), \tag{61}$$

$$\sigma'_{ij,21} = \sum_{k=1}^{n_a}\sum_{l=1}^{2}\sum_{h=1}^{2} w'_k(x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j), \tag{62}$$

$$\sigma'_{ij,22} = \sum_{k=1}^{n_a}\sum_{l=1}^{2} w'_k(x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j), \tag{63}$$

$$\sigma'_{ij,31} = \sum_{k=1}^{n_d}\sum_{l=1}^{2}\sum_{h=1}^{2} w'_k(x_{i,l,k} - \mu'_i)(x_{j,h,k} - \mu'_j), \tag{64}$$

$$\sigma'_{ij,32} = \sum_{k=1}^{n_d}\sum_{l=1}^{3} w'_k(x_{i,l,k} - \mu'_i)(x_{j,l,k} - \mu'_j). \tag{65}$$

Plugging-in the values of μ'_i and μ'_j in (60), we obtain:

$$\begin{aligned}
\sigma'_{ij,11} &= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{2}\sum_{h=1}^{2} w'_k(x_{i,l,k} - \frac{s}{s'}\mu_i - \mu_{i,a} + \mu_{i,d})(x_{j,h,k} - \frac{s}{s'}\mu_j - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{2}\sum_{h=1}^{2} w'_k(x_{i,l,k} - \mu_i + \mu_i(1 - \frac{s}{s'}) - \mu_{i,a} + \mu_{i,d}) \\
&\qquad (x_{j,h,k} - \mu_j + \mu_j(1 - \frac{s}{s'}) - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{2}\sum_{h=1}^{2} w'_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{2}\sum_{h=1}^{2} w'_k(x_{i,l,k} - \mu_i)(\mu_j(1 - \frac{s}{s'}) - \mu_{j,a} + \mu_{j,d}) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{2}\sum_{h=1}^{2} w'_k(\mu_i(1 - \frac{s}{s'}) - \mu_{i,a} + \mu_{i,d})(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{2}\sum_{h=1}^{2} w'_k(\mu_i(1 - \frac{s}{s'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \frac{s}{s'}) - \mu_{j,a} + \mu_{j,d}).
\end{aligned} \tag{66}$$

Since $\sum_{k=1}^{n}\sum_{l=1}^{2} w'_k(x_{i,l,k} - \mu_i) = 0$, $1 \le i \le 2$, we have

$$\begin{aligned}
\sigma'_{ij,11} &= \textstyle\frac{1}{s'}\sum_{k=1}^{n}\sum_{l=1}^{2}\sum_{h=1}^{2} s_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\frac{1}{s'}\sum_{k=1}^{n}\sum_{l=1}^{2}\sum_{h=1}^{2} s_k(\mu_i(1 - \frac{s}{s'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \frac{s}{s'}) - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\frac{1}{s'}\sum_{k=1}^{n}\sum_{l=1}^{2}\sum_{h=1}^{2} s_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle 4\frac{s}{s'}(\mu_i(1 - \frac{s}{s'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \frac{s}{s'}) - \mu_{j,a} + \mu_{j,d}).
\end{aligned} \tag{67}$$

Plugging-in the values of μ'_i and μ'_j in (61), we obtain:

$$
\begin{aligned}
\sigma'_{ij,12} &= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{2} w'_k(x_{i,l,k} - \frac{s}{s'}\mu_i - \mu_{i,a} + \mu_{i,d})(x_{j,h,k} - \frac{s}{s'}\mu_j - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{2} w'_k(x_{i,l,k} - \mu_i + \mu_i(1 - \frac{s}{s'}) - \mu_{i,a} + \mu_{i,d}) \\
&\qquad\qquad (x_{j,h,k} - \mu_j + \mu_j(1 - \frac{s}{s'}) - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\sum_{k=1}^{n}\sum_{l=1}^{2} w'_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{2} w'_k(x_{i,l,k} - \mu_i)(\mu_j(1 - \frac{s}{s'}) - \mu_{j,a} + \mu_{j,d}) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{2} w'_k(\mu_i(1 - \frac{s}{s'}) - \mu_{i,a} + \mu_{i,d})(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\sum_{k=1}^{n}\sum_{l=1}^{2} w'_k(\mu_i(1 - \frac{s}{s'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \frac{s}{s'}) - \mu_{j,a} + \mu_{j,d}).
\end{aligned}
\tag{68}
$$

Since $\sum_{k=1}^{n}\sum_{l=1}^{2} w'_k(x_{i,l,k} - \mu_i) = 0$, $1 \le i \le 2$, we have

$$
\begin{aligned}
\sigma'_{ij,12} &= \textstyle\frac{1}{s'}\sum_{k=1}^{n}\sum_{l=1}^{2} s_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle\frac{1}{s'}\sum_{k=1}^{n}\sum_{l=1}^{2} s_k(\mu_i(1 - \frac{s}{s'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \frac{s}{s'}) - \mu_{j,a} + \mu_{j,d}) \\
&= \textstyle\frac{1}{s'}\sum_{k=1}^{n}\sum_{l=1}^{2} s_k(x_{i,l,k} - \mu_i)(x_{j,h,k} - \mu_j) + \\
&\quad \textstyle 2\frac{s}{s'}(\mu_i(1 - \frac{s}{s'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \frac{s}{s'}) - \mu_{j,a} + \mu_{j,d}).
\end{aligned}
\tag{69}
$$

From (68) and (69), we obtain

$$
\begin{aligned}
\sigma'_{ij,1} &= \sigma'_{ij,11} + \sigma'_{ij,12} \\
&= \sigma_{ij} + 6\tfrac{s}{s'}(\mu_i(1 - \tfrac{s}{s'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \tfrac{s}{s'}) - \mu_{j,a} + \mu_{j,d}).
\end{aligned}
\tag{70}
$$

Note that $\vec{\mu}'$ and the components $\sigma'_{ij,21}$, $\sigma'_{ij,22}$, $\sigma'_{i,31}$ and $\sigma'_{i,32}$ can be computed in constant time, under the assumption that X and X' differ in the constant number of polyhedra, i.e., n_a and n_d are constants. In that case, also the component

$$
\begin{aligned}
\sigma'_{ij} &= \tfrac{1}{6}(\sigma'_{ij,11} + \sigma'_{ij,12} + \sigma'_{ij,21} + \sigma'_{ij,22} + \sigma'_{ij,31} + \sigma'_{ij,32}) \\
&= \tfrac{1}{6}(\sigma_{ij} + \sigma'_{ij,21} + \sigma'_{ij,22} + \sigma'_{ij,31} + \sigma_{ij,32}) + \\
&\quad \tfrac{s}{s'}(\mu_i(1 - \tfrac{s}{s'}) - \mu_{i,a} + \mu_{i,d})(\mu_j(1 - \tfrac{s}{s'}) - \mu_{j,a} + \mu_{j,d})
\end{aligned}
\tag{71}
$$

can be computed in $O(n_a + n_d)$ time.

Deleting points

Let the new convex hull be obtained by deleting n_d tetrahedra from and added n_a tetrahedra to the old convex hull. Consequently, the same formulas and time complexity, as by adding points, follow.